TUGAS AKHIR – KI1502

RANCANG BANGUN RANDOM WORD GENERATOR DENGAN ALGORITMA BACKTRACKING PADA GAMEPLAY UNTUK GAME AMBROSIA

FAJAR SETIAWAN
NRP 5112100010

Dosen Pembimbing
Imam Kuswardayan, S.Kom., M.T.
Dr. Eng. Nanik Suciati, S.Kom., M.Kom.

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2016

(Halaman ini sengaja dikosongkan)

FINAL PROJECT– KI1502

DESIGN AND IMPLEMENTATION OF RANDOM WORD GENERATOR USING BACKTRACKING ALGORITHM FOR GAMEPLAY IN AMBROSIA GAME

FAJAR SETIAWAN
NRP 5112100010

Advisor
Imam Kuswardayan, S.Kom., M.T.
Dr. Eng. Nanik Suciati, S.Kom., M.Kom.

INFORMATICS DEPARTMENT
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2016

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

RANCANG BANGUN RANDOM WORD GENERATOR DENGAN ALGORITMA BACKTRACKING PADA GAMEPLAY UNTUK GAME AMBROSIA

Tugas Akhir

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Rumpun Mata Kuliah Interaksi, Grafika, dan Seni
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
FAJAR SETIAWAN
NRP. 5112 100 010

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Imam Kuswardayan, S.Kom., M.T. .....
NIP: 19761215 200312 1 001 (pembimbing 1)

Dr. Eng. Nanik Suciati, S.Kom., M.Kom. .....
NIP: 19710428 199412 2 001 (pembimbing 2)

SURABAYA
JUNI, 2016

(Halaman ini sengaja dikosongkan)

RANCANG BANGUN RANDOM WORD GENERATOR DENGAN ALGORITMA BACKTRACKING PADA GAMEPLAY UNTUK GAME AMBROSIA

Nama Mahasiswa : Fajar Setiawan
NRP : 5112 100 010
Jurusan : Teknik Informatika FTIf-ITS
Dosen Pembimbing I : Imam Kuswardayan, S.Kom., M.T.
Dosen Pembimbing II : Dr. Eng. Nanik Suciati, S.Kom., M.Kom.

ABSTRAK

Bermain game merupakan salah satu cara untuk mengisi waktu luang. Game sendiri diciptakan untuk menghibur pengguna yang memainkannya. Salah satu genre game yang sangat menghibur dan menyenangkan serta merupakan favorit dari pecinta game adalah Role Playing Game (RPG). Game RPG sangat disukai karena pemain dapat mengkostumasi dan mengembangkan karakter dalam game tersebut sesuai dengan yang diinginkan oleh pemain.

Perkembangan game menuntut inovasi baru yang terus hadir untuk menyesuaikan kebutuhan pasar. Salah satu inovasi dalam game terletak pada gameplay. Dalam game RPG yang menganut sistem Turn Base Strategy (TBS), improvisasi gameplay sangatlah minim. Gameplay pada sistem TBS lebih mengedepankan strategi, sedangkan untuk aksi tiap pemain sangat kecil, sehingga membuat beberapa pemain awam akan merasa bosan di awal. Oleh karena itu, untuk mengatasi masalah tersebut, maka dikembangkan pembaruan terhadap gameplay TBS.

Ambrosia merupakan game RPG dengan improvisasi pada gameplay TBS. Improvisasi yang dilakukan berupa penyusunan kata dari huruf acak (anagram) untuk menyerang musuh. Proses pengacakan dan pemilihan kata dilakukan dengan mempertimbangkan bobot huruf awal. Sedangkan dalam

membentuk solusi kata yang dapat terbentuk menggunakan Algoritma Backtracking.

Hasil pengujian dari tugas akhir ini menunjukkan bahwa Algoritma Backtracking yang digunakan mampu menyusun solusi kata. Selain itu, improvisasi gameplay yang dilakukan berhasil menarik perhatian pengguna untuk memainkan Ambrosia lagi.

Kata kunci**: **Role Playing Game (RPG), Turn Base Strategy (TBS), Improvisasi, Anagram, Algoritma Backtracking.

DESIGN AND IMPLEMENTATION OF RANDOM WORD GENERATOR USING BACKTRACKING ALGORITHM FOR GAMEPLAY IN AMBROSIA GAME

Student Name	: Fajar Setiawan
NRP	: 5112 100 010
Major	: Teknik Informatika FTIf-ITS
Advisor I	: Imam Kuswardayan, S.Kom., M.T.
Advisor II	: Dr. Eng. Nanik Suciati, S.Kom., M.Kom.

ABSTRACT

Playing games is one way to spend our free time. Game was created to entertain users who play it. One favorite genre for some gamers is a Role Playing Game (RPG). RPG is favored because players assume the roles of characters in a fictional setting.

Game development requires new innovations to fulfill the market's requirement. One of the innovations in the game lies in the gameplay. In the RPG game that adopts Turn Base Strategy (TBS), improvised gameplay is very minimal. TBS system is focus in strategy, while for the action of each player is very little, that makes some novice players feel bored. Therefore, to solve these problems, the TBS gameplay need to improved.

Ambrosia is RPG game with TBS gameplay that improved. The improvement gameplay is creating a word from random letters (Anagram) to attack the enemy. The randomization process is considering by the weight of the initial letter. And for extraction of words solution that can created is using Backtracking Algorithm.

After testing this game, it shows that words solution can extracted by using Backtracking Algorithm. The improved gameplay also successfully attract users to play Ambrosia again.

Keywords**: **Role Playing Game (RPG), Turn Base Strategy (TBS), Improvisation, Anagram, Backtracking Algorithm.

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

Segala puji dan syukur kehadirat Allah Subhanahu wa Ta'ala yang telah memberikan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul "RANCANG BANGUN RANDOM WORD GENERATOR DENGAN ALGORITMA BACKTRACKING PADA GAMEPLAY UNTUK GAME AMBROSIA".

Pengerjaan tugas akhir ini dilakukakn untuk memenuhi salah satu syarat memeroleh gelar Sarjana Komputer di Program Studi S-1 Jurusan Teknik Informatika Fakultas Teknologi Informasi Institut Teknologi Sepuluh Nopember. Harapan penulis apa yang dikembangkan dalam tugas akhir ini dapat bermanfaat bagi banyak kalangan.

Selama pelaksaanan dan pembuatan tugas akhir ini, penulis mengucapkan rasa terima kasih kepada semua pihak yang membantu dan memberikan semangat kepada penulis baik secara langsung maupun tidak, antara lain:

1. Allah SWT. atas limpahan rahmat dan rezeki-Nya sehingga penulis dapat menyelesaikan tugas akhir. Serta Nabi Muhammad SAW atas jasa beliau, menjadi tauladan kita dalam segala hal, termasuk pengerjaan tugas akhir ini.
2. Keluarga penulis, Ibu Ernawati Pantjarini, Bapak Joko Suroso, kakak Farid Alfianto, serta keluarga yang tidak dapat penulis sebut satu persatu yang telah memberikan banyak dukungan moral, doa, dan materil untuk penulis.
3. Bapak Imam Kuswadaryan selaku dosen pembimbing tugas akhir pertama dan yang telah memberikan arahan dalam pengerjaan tugas akhir ini.
4. Ibu Nanik Suciati selaku dosen pembimbing tugas akhir kedua dan juga selaku dosen wali yang dengan sabar membimbing penulis dalam pengerjaan tugas akhir ini.
5. Dosen-dosen, staf dan karyawan Teknik Informatika yang dengan sabar mendidik dan memberikan pengalaman baru kepada penulis selama berkuliah di Teknik Informatika.

6. Sahabat-sahabat penulis yang telah banyak membantu penulis sejak awal perkuliahan hingga mengerjakan tugas akhir ini.
7. Rekan-rekan Laboratorium Interaksi, Grafika, dan Seni yang saling membantu satu sama lain.
8. Pihak-pihak lain yang tidak bisa penulis sebut satu persatu yang juga berjasa dalam membantu penulis mengerjakan tugas akhir.

Penulis telah berusaha sebaik mungkin dalam menyusun Tugas Akhir ini, namun penulis memohon maaf apabila terdapat kekurangan, kesalahan maupun kelalaian dalam pengerjaannya. Kritik dan saran yang membangun dapat disampaikan kepada penulis sebagai bahan perbaikan selanjutnya.

Surabaya, Juni 2016
Penulis

Fajar Setiawan

DAFTAR ISI

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

DAFTAR KODE SUMBER

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

Bab ini memaparkan garis besar Tugas Akhir yang meliputi latar belakang, tujuan dan manfaat pembuatan, rumusan dan batasan permasalahan, metodologi pembuatan Tugas Akhir, dan sistematika penulisan.

1.1. Latar Belakang

Bermain *game* merupakan salah satu cara untuk mengisi waktu luang. *Game* sendiri diciptakan untuk menghibur pengguna yang memainkannya. Salah satu *genre game* yang sangat menghibur dan menyenangkan serta merupakan favorit dari pecinta *game* adalah *Role Playing Game (RPG)*. *Game RPG* sangat disukai karena pemain dapat mengkostumasi dan mengembangkan karakter dalam game tersebut sesuai dengan yang diinginkan oleh pemain. Salah satu *game RPG* yang cukup populer adalah *Pokemon Series* yang dikembangkan oleh Nintendo [1].

Perkembangan *game* yang semakin pesat sejalan dengan pertumbuhan masyarakat yang menyukai *game*. Semua lapisan usia sudah dapat dicakup berbagai macam *game* yang ada di pasar. Namun, diperlukan inovasi baru yang terus hadir untuk menyesuaikan kebutuhan pasar. Salah satu inovasi dalam game terletak pada *gameplay*. Dalam *game RPG* yang menganut sistem *Turn Base Strategy* (*TBS*), improvisasi *gameplay* sangatlah minim. *Gameplay* pada sistem *TBS* lebih mengedepankan strategi, sedangkan untuk aksi tiap pemain sangat kecil, sehingga membuat beberapa pemain awam akan merasa bosan di awal. Oleh karena itu, untuk mengatasi masalah tersebut, maka dikembangkan pembaruan terhadap *gameplay TBS*.

Improvisasi yang akan menjadi fokus dari pengajuan Tugas Akhir ini adalah *gameplay* baru berupa penyusunan kata dari huruf yang diacak (anagram) untuk menyerang musuh. Dalam *gameplay* ini, pemain harus menyusun sebuah kata sepanjang mungkin untuk menyerang musuh. Semakin panjang kata yang disusun, maka

semakin besar serangan yang diberikan. Kata yang dapat pemain susun adalah kata dalam bahasa Inggris. Terdapat 2 komponen utama dalam *gameplay* ini, yang pertama adalah pengacakan huruf, dan yang kedua adalah pengecekan kata. Untuk pengacakan huruf, akan dipilih kata dari *database* atau kamus referensi bahasa Inggris. Kemudian pada pengecekan kata, dibagi menjadi 2 tahap, yakni pembuatan list kata dan pengecekan list kata. List kata merupakan kumpulan kata yang dapat disusun dari huruf yang sebelumnya diacak dan terdapat di kamus bahasa Inggris. Pembuatan list kata merupakan implementasi dari Algoritma *Backtracking*. Kemudian, setelah terbentuk list, *game* akan menunggu *input* kata dari pemain. Ketika kata dari pemain sudah tersusun, maka akan dicek dengan list yang sudah terbentuk. Pengecekan ini menggunakan Algoritma *Brute Force*.

Dengan pembaruan *gameplay* ini, diharapkan bagi pengguna awam yang belum pernah memainkan *game RPG* tidak merasa bosan dan jenuh. Selain itu, *game* ini diharapkan dapat melatih kemampuan berbahasa Inggris dari pemain.

1.2. Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut:

1. Bagaimana mancari kata secara acak berdasarkan huruf awal yang berasal dari kamus referensi.
2. Bagaimana penerapan *Algoritma Backtracking* dalam menyusun solusi kata Bahasa Inggris yang tepat.
3. Bagaimana implementasi dari rumusan diatas diselesaikan dengan *Game Engine Unity*.

1.3. Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, di antaranya sebagai berikut:

1. Kombinasi huruf yang digunakan dibatasi maksimal tujuh huruf dan minimal tiga huruf.

2. Menggunakan sebuah kamus bahasa Inggris yang disediakan oleh Infochimps sebagai referensi untuk pengecekan kata.

1.4. Tujuan

Tujuan dari pembuatan Tugas Akhir ini adalah:

1. Menerapkan inovasi baru dalam *gameplay game RPG* yang menerapkan sistem *TBS*, berupa penyusunan kata dari huruf acak untuk menyerang musuh agar terdapat aksi.
2. Mengasah kemampuan berbahasa Inggris pemain.
3. Mengenalkan *game RPG* yang interaktif pada pemain awam.

1.5. Manfaat

Manfaat dari hasil pembuatan Tugas Akhir ini antara lain:

1. Mengajarkan pengguna kata-kata dalam Bahasa Inggris.
2. Mengasah kemampuan dan kecepatan berpikir pengguna dalam menyusun kata.
3. Penerapan improvisasi dari *gameplay* pada tugas akhir ini dapat dikembangkan pada *game* lainnya.
4. Sebagai sarana hiburan untuk para pengguna.

1.6. Metodologi

Pembuatan Tugas Akhir dilakukan menggunakan metodologi sebagai berikut:

A. Studi literature
   Pada studi literatur ini, akan dipelajari sejumlah referensi yang diperlukan dalam pembuatan aplikasi yaitu mengenai Rancang Bangun Perangkat Lunak, *Game Engine* Unity3D, Android SDK, Algoritma *Backtracking*, Algoritma *Brute Force*, Algoritma Seleksi Mesin *Roulette*, dan Code::Blocks.

B. Perancangan perangkat lunak
   Analisis dan perancangan dalam pembuatan *Game Ambrosia* antara lain sebagai berikut :
   1. Analisis permasalahan dan deskripsi umum

2. Analisis kebutuhan
3. Analisis karakter pengguna
4. Perancangan alur permainan
5. Perancangan algoritma

C. Implementasi dan pembuatan sistem

Game ini dibangun dengan menggunakan bahasa C# pada *Unity3D*. Selain itu, dibutuhkan kamus referensi atau *dictionary* untuk database kata. Untuk proses filterisasi pada kamus referensi menggunakan bahasa C++ pada CodeBlocks. *Game* yang dikembangkan berbasis android.

D. Uji coba dan evaluasi

Pengujian dari *game* ini dilakukan dengan beberapa cara, antara lain :

1. Pengujian *blackbox.*
   Pengujian ini berfokus pada pengujian fungsional dari sistem yang dikembangkan. Pengujian ini dilakukan untuk mengetahui apakah kinerja permainan sudah sesuai dengan kebutuhan atau tidak.
2. Pengujian usabilitas.
   Pengujian usabilitas dilakukan dengan cara melakukan survei ke beberapa pengguna yang gemar bermain *game RPG* serta pengguna yang jarang bermain *game RPG* di sekitar lingkungan Teknik Informatika ITS. Survei dilakukan untuk mengukur tingkat kegunaan dari aplikasi yang dibuat dalam membantu pengguna. [2]

E. Penyusunan laporan tugas akhir

Pada tahap ini, dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam Tugas Akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat.

1.7. Sistematika Penulisan

Buku Tugas Akhir ini terdiri dari beberapa bab, antara lain sebagai berikut.

BAB I PENDAHULUAN

Bab ini berisi latar belakang masalah, rumusan dan batasan permasalahan, tujuan dan manfaat pembuatan tugas akhir, metodologi yang digunakan, dan sistematika penyusunan tugas akhir.

BAB II TINJAUAN PUSTAKA

Bab ini membahas dasar pembuatan dan beberapa teori penunjang yang berhubungan dengan pokok pembahasan yang mendasari pembuatan tugas akhir ini.

BAB III ANALISIS DAN PERANCANGAN SISTEM

Bab ini membahas analisis dari sistem yang dibuat meliputi analisis permasalahan, deskripsi umum perangkat lunak, spesifikasi kebutuhan, dan identifikasi pengguna. Kemudian membahas rancangan dari sistem yang dibuat meliputi rancangan skenario kasus penggunaan, arsitektur, data, dan antarmuka.

BAB IV IMPLEMENTASI

Bab ini membahas implementasi dari rancangan sistem yang dilakukan pada tahap perancangan. Penjelasan implementasi meliputi implementasi pembangkitan area permainan, dan antarmuka permainan.

BAB V PENGUJIAN DAN EVALUASI

Bab ini membahas pengujian dari aplikasi yang dibuat dengan melihat keluaran yang dihasilkan oleh aplikasi dan evaluasi untuk mengetahui kemampuan aplikasi.

BAB VI PENUTUP

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan serta saran untuk pengembangan aplikasi selanjutnya.

(Halaman ini sengaja dikosongkan)

BAB II
TINJAUAN PUSTAKA

Pada bab ini akan dibahas mengenai teori-teori yang menjadi dasar dari pembuatan Tugas Akhir ini. Teori-teori tersebut *adalah Unity3D,* Rancang Bangun Perangkat Lunak, Android SDK, Algoritma *Backtracking,* Algoritma *Brute Force,* Algoritma *Roulette Wheel Selection,* dan Code::Blocks.

2.1 Unity3D Game Engine

Unity merupakan sebuah *software Developing Game* yang berbasis *MultiPlatform* [3]. Unity adalah *tool* pengembang video *game* yang membantu orang mengembangkan *game enviroiment* 3D maupun 2D. Lingkungan pengembang unity berjalan di Microsoft Windows dan Mac OS, *game* yang dihasilkan dapat dijalankan di Windows, Mac, Xbox 360, PlayStation 3, Wii, iPad, Iphone, dan Android. Unity juga bisa menghasilkan permainan di *browser* dengan menggunakan *plugin* Unity Web Player, bisa digunakan untuk Windows dan Mac tetapi belum kompatibel untuk Linux. Unity perlu lisensi untuk dapat di-*publish* ke platform tertentu, seperti format pada tipe format ".apk" Tetapi Unity menyediakan untuk free user dan bisa di *publish* dalam bentuk *Standalone* (.exe) dan web. Untuk saat ini Unity sedang mengembangkan *software* berbasis *Augment Reality* (AR) dan *Virtual Reality* (VR).

2.2 Rancang Bangun Perangkat Lunak

Rancangan sistem adalah penentuan proses dan data yang diperlukan oleh sistem baru. Bangun sistem adalah membangun sistem informasi dan komponen yang didasarkan pada spesifikasi desain.

Dengan demikian pengertian rancang bangun merupakan kegiatan menerjemahkan hasil analisa ke dalam bentuk paket perangkat lunak kemudian menciptakan sistem tersebut [4]. Dalam merancang perangkat lunak terdapat model dan struktur data yang

digunakan oleh sistem, antarmuka antar komponen dan sitem, serta algoritma yang dugunakan.

2.3 Android SDK

Android-SDK merupakan tools bagi para *programmer* yang ingin mengembangkan aplikasi berbasis google android. Android SDK mencakup seperangkat alat pengembangan yang komprehensif. Android SDK terdiri dari *debugger*, *libraries*, *handset emulator*,dokumentasi, contoh kode, dan tutorial. Saat ini Android sudah mendukung arsitektur x86 pada Linux (distribusi Linux apapun untuk desktop modern), Mac OS X 10.4.8 atau lebih, Windows XP, 7, 8, dan 8.1. Persyaratan mencakup JDK, Apache Ant dan Python 2.2 atau yang lebih baru. IDE yang didukung secara resmi adalah Eclipse 3.2 atau lebih dengan menggunakan *plugin Android Development* Tools (ADT), dengan ini pengembang dapat menggunakan teks editor untuk mengedit file Java dan XML serta menggunakan peralatan *command line* untuk menciptakan, membangun, melakukan *debug* aplikasi Android dan pengendalian perangkat Android (misalnya, *reboot*, menginstal paket perangkat lunak dengan jarak jauh) [5].

2.4 Algoritma *Backtracking*

Backtracking adalah algoritma berbasis *Depth First Search* (DFS) untuk mencari solusi suatu persoalan. Backtracking merupakan algoritma perbaikan dari *brute-force*, yang secara sistematis hanya mencari solusi yang mungkin [6]. Umumnya Algoritma *Backtracking* bersifat rekursif, namun ada pula versi *backtracking* yang iteratif. Langkah-langkah pencarian solusi adalah sebagai berikut :

1. Solusi dicari dengan membentuk lintasan dari akar ke daun. Simpul yang dilahirkan bernama simpul hidup.
2. Jika lintasan yang sedang dibentuk tidak mengarah ke solusi maka simpul hidup dibunuh menjadi simpul mati. Fungsi yang digunakan untuk membunuh fungsi adalah fungsi pembatas.

3. Jika pembentukan simpul hidup terakhir adalah simpul mati maka proses pencarian diteruskan dengan membangkitkan simpul anak yang lain. Bila tidak ada simpul anak lagi maka pencarian melakukan *backtracking* ke simpul orangtuanya.
4. Pencarian berakhir bila ditemukan solusi atau tidak ada lagi simpul yang hidup untuk *backtrack*.

2.5 Algoritma *Brute Force*

Brute force adalah sebuah pendekatan yang langsung (*straightforward*) untuk memecahkan suatu masalah, biasanya didasarkan pada pernyataan masalah (*problem statement*) dan definisi konsep yang dilibatkan [7]. Algoritma brute force memecahkan masalah dengan sangat sederhana, langsung dan dengan cara yang jelas (*obvious way*). Langkah yang dilakukan dalam Algoritma adalah memproses data satu persatu sebanyak *n* data sebagai kemungkinan terburuknya.

2.6 Algoritma *Roulette Wheel Selection*

Algoritma *Roulette Wheel Selection* merupakan metode seleksi berdasarkan besarnya nilai dari suatu objek [8]. Metode ini menggunakan persentase dari tiap objek. Besarnya persentase tiap objek menyesuaikan dengan nilainya. Semakin besar persentasenya, semakin besar objek tersebut terpilih. Konsep dari metode seleksi ini seperti mesin putar *Roulette*. Langkah dalam proses seleksi algoritma ini adalah:

1. Tentukan persentase atau rentang dari tiap objek.
2. Acak nilai dari nilai rentang minimal hingga rentang akhir.
3. Jika hasil pengcakan berada pada rentang suatu objek, maka objek tersebut yang terpilih.

2.7 Code::Blocks

Code::Blocks merupakan *free Integrated Development Environment* (*IDE*) yang terbuka dan *multi platform*. Program

yang ditulis dalam C++ beserta wxWidgets untuk *Graphical User Interface* (*GUI*) ini bisa digunakan bersama dengan berbagai macam kompilator, seperti GCC dan Visual C++. *Tools* yang tersedia tergantung dari *plugin* yang terpasang. Saat ini, Code::Blocks tersedia sebagai *IDE* untuk bahasa C dan C++, walaupun program ini juga bisa disesuaikan, dan mungkin akan membutuhkan pemasangan tambahan, untuk pengembangan perangkat lunak [9].

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Bab ini menjelaskan analisis permasalahan dan perancangan dari *game* yang akan dibangun. Analisis permasalahan membahas permasalahan yang yang diangkat dalam tugas akhir. Analisis kebutuhan mencantumkan kebutuhan-kebutuhan yang diperlukan pada permainan. Selain itu akan dijabarkan pula deskripsi dari permainan yang akan dikembangkan. Untuk tahap perancangan akan menjabarkan perancangan permainan.

3.1 Analisis Sistem

Tahap analisis dibagi menjadi beberapa bagian antara lain analisis permasalahan, deskripsi umum aplikasi, analisis kebutuhan, dan karateristik pengguna.

3.1.1 Analisis Permasalahan

Permasalahan yang diangkat pada tugas akhir ini adalah sedikitnya inovasi dalam sistem bertarung pada permainan dengan *genre RPG* dengan mekanisme *TBS*. Kebanyakan pembaruan pada permainan berjenis tersebut hanya terdapat pada *story line* atau jenis musuh yang semakin bervariasi dan kostumasi karakter yang semakin leluasa. Terlebih, permainan dengan mekanisme *TBS* sangat minim aksi, sehingga pemain awam akan merasa bosan di awal.

Solusi yang sudah ada untuk mengatasi permasalahan ini adalah dengan memberikan durasi pada saat bertarung sehingga pemain akan lebih terpacu dan tidak terlalu lama memikirkan sesuatu yang lama kelamaan akan memberi rasa bosan. Akan tetapi, solusi ini belum menyelesaikan semua permalahan diatas. Pemain yang sudah terbiasa dengan sistem durasi akan melakukan hal yang sama seperti menyerang musuh dengan salah satu jurus atau *skill* berulang kali (jika syarat memenuhi) tanpa memerdulikan durasi. Tentu saja hal ini akan semakin membosankan untuk pengguna awam.

Oleh karena itu, diperlukan suatu solusi, dimana pemain tidak bisa menyerang musuh dengan cara yang sama berulang kali dan

berpikir dalam waktu yang terbatas untuk mengalahkan musuh yang dihadapi. Permainan yang akan dibuat dalam tugas akhir ini akan menjawab permasalahan diatas dengan memasukkan unsur edukasi berupa bahasa Inggris pada sistem bertarungnya. Diharapkan dengan adanya aplikasi ini dapat membantu pengenalan *game RPG* pada pengguna awam serta melatih kemampuan pemain dalam menggunakan kosa-kata bahasa Inggris.

3.1.2 Deskripsi Umum Aplikasi

Aplikasi atau permainan yang akan dibangun dalam tugas akhir ini merupakan permainan dengan *genre RPG* dengan mekanisme bertarung *TBS* yang dikombinasikan dengan unsur edukasi berupa penggunaan kata dalam bahasa Inggris. *Game* ini bercerita tentang seseorang yang ingin mencari *Ambrosia*, yakni makanan untuk para dewa-dewi. Dengan memakan *Ambrosia*, maka seseorang tersebut dapat memiliki kekuatan setara dewa dan dapat mengalahkan dewa yang merusak dunia.

Fokus utama dari *game* ini terletak pada mekanisme bertarung atau *gameplay*. Pemain akan dihadapkan dengan kumpulan huruf, lalu pemain diminta untuk menyusun suatu kata dalam bahasa Inggris dengan benar untuk menyerang musuh. Semakin panjang kata yang bisa dibentuk, maka semakin besar pula nilai serangan atau *damage* yang diberikan kepada musuh. Pada sistem *game* ini, huruf acak yang ditampilkan kepada pemain berasal dari kata acak yang diambil dari kamus referensi, kemudian kata acak tersebut dicari kemungkinan kata yang bisa dibentuk dengan menggunakan Algoritma *Backtraking*. Setelah mendapat list kata yang dibentuk, maka *input* kata yang diberikan pemain tinggal mencocokkan dengan list menggunakan *Brute Force.*

Dalam permainan ini akan disiapkan beberapa *dungeon* yang bisa dimainkan. Karena ini merupakan *game RPG*, maka pemain dapat melakukan pembaruan dan kostumasi. Hanya saja, ada beberapa batasan dalam melakukan kostumasi, seperti hanya menambah atribut tapi tidak merubah penampilan. Pada permainan ini, pemain dapat meningkatkan level karakter mereka hingga level 20. Untuk

pengembangan karakter dan kesulitan dalam melawan musuh diatur otomatis oleh sistem.

3.1.3 Analisis Kebutuhan

Kebutuhan utama dari permainan ini adalah sistem mampu memberikan solusi berupa kumpulan kata yang dapat dibentuk dari kata yang telah terpilih secara acak. Adapun kebutuhan fungsional dari permainan ini dapat dilihat pada Tabel 3.1.

Tabel 3.1 Spesifikasi Kebutuhan Fungsional

Kode Kebutuhan	Kebutuhan Fungsional	Deskripsi
F-0001	Menyusun Kata	Pemain dapat menyusun kata dari huruf acak yang disediakan sistem
F-0002	Mencari solusi dari kata yang diacak	Sistem dapat memberikan solusi berupa kumpulan kata yang dapat terbentuk sehingga dapat dicocokkan dengan *input* pemain
F-0003	Menggunakan *skill* dan *potion*	Terdapat *skill* yang bisa digunakan oleh pemain serta *potion* untuk membatu memenangkan pertarungan
F-0004	Memilih *dungeon*	Pemain dapat memilih *dungeon* dan melihat syarat untuk menyelesaikan *dungeon* tersebut
F-0005	Menjelajahi *dungeon*	Pemain dapat berjalan untuk memutari *dungeon* agar dapat bertemu dengan musuh
F-0006	Melihat progres level dan uang	Terdapat progres level saat ini dan juga uang yang dimiliki untuk.
F-0007	Melihat Status Karakter	Terdapat status karakter berupa *attack, speed, hp, mp*, senjata, armor, serta elemen yang dipilih
F-0008	Melihat Keterangan Senjata dan Armor	Pemain dapat melihat keterangan dari senjata atau armor yang digunakan

F-0009	Melihat *item* yang dimiliki	Pemain dapat melihat jumlah item apa saja yang dimiliki saat ini
F-0010	Membeli *item*	Pemain dapat membeli *item* dengan jumlah yang pemain inginkan
F-0011	Memerbarui senjata, dan armor	Pemain dapat memerbarui senjata atau armor yang dikenakan.
F-0012	Memerbarui *skill*	Pemain dapat memerbarui *skill-skill* yang dimiliki berdasar elemen yang dipilih
F-0013	Menaikkan Level	Pemain dapat meningkatkan level dan status setelah mendapat *EXP* dari musuh yang dikalahkan. Sistem kemudian akan menset EXP untuk level selanjutnya
F-0014	Mengalahkan Musuh	Pemain dapat mengalahkan musuh jika *HP* musuh kurang dari 0

3.1.4 Karakteristik Pengguna

Berdasarkan deskripsi dan permasalahan yang telah dijabarkan, diperoleh karekteristik pengguna yang dapat menggunakan aplikasi ini. Karakteristik tersebut dapat dilihat pada Tabel 3.2

Tabel 3.2 Karakteristik Pengguna

Nama Aktor	Tugas	Hak Akses Aplikasi	Kemampuan yang harus dimiliki
Pemain	Memainkan permainan mulai dari tutorial	Dapat menjalankan segala fungsionalitas dalam permainan	Memiliki kemampuan dasar dalam berbahasa Inggris

3.2 Perancangan Permainan

Tahap perancangan dibagi menjadi beberapa bagian, antara lain perancangan alur permainan, perancangan antarmuka, serta perancangan algoritma.

3.2.1 Perancangan Alur Permainan

Alur permainan merupakan serangkaian proses yang harus diikuti oleh pemain dalam memainkan permainan ini. Rancangan alur dalam permainan ini menggunakan pemodelan *Finite State Machine (FSM). Game* ini bercerita pada suatu masa di dunia pararel, dimana tiba-tiba para dewa turun ke bumi dan mengacaukan dunia. Kekuatan manusia tidak sebanding dengan kekuatan yang dimiliki oleh dewa. Hingga suatu hari, datanglah seseorang yang ingin mencari *Ambrosia*, yakni makanan untuk para dewa-dewi. Dengan memakan *Ambrosia*, maka seseorang tersebut dapat memiliki kekuatan setara dewa dan dapat mengalahkan dewa yang merusak dunia. Pemain akan berperan sebagai seseorang tersebut.

Pada awal permainan, pemain akan menikmati cerita asal mula kejadian ini terjadi dan siapa sebenarnya tokoh utama dalam cerita ini. Selama bermain, pemain akan menjalani tutorial yang diselingi dengan cerita agar semakin menarik. Tutorial hanya dijalankan sekali saja selama memainkan *game* ini.

Pada *panel main menu*, pemain dapat melihat progres level beserta *gold* yang dimiliki. Selain itu, pemain juga dapat membuka 4 *panel* berupa *dungeon panel, arsenal panel, foundry panel,* serta *other panel*. Sebagai *default*, *panel* yang aktif pertama kali ketika pemain masuk *main menu* adalah *dungeon panel*. Untuk memulai permainan, pemain perlu berada pada *dungeon panel*, kemudian pemain memilih *stage* yang diinginkan. Pemain juga dapat memilih *stage* yang berada pada *dungeon map* yang berbeda dengan menekan tombol *next/previous*. Terdapat 3 *dungeon map* yang bisa dipilih. Setelah memilih *stage* yang diinginkan, pemain dapat melihat *task list* yang harus diselesaikan selama menaklukkan *dungeon*. *Task list* yang diberikan berupa jumlah musuh yang harus dilawan selama berada di *dungeon*. Jika pemain merasa sanggup, maka pemain dapat masuk ke

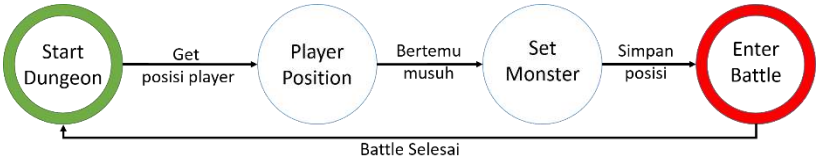
dalam *dungeon* tersebut. Untuk *task list* setiap *stage* dapat dilihat pada Tabel 3.3.

Tabel 3.3 *Task List per Stage*

Stage	***Task List***
1/Tutorial	- Mengalahkan 1 Mandrago-Witch
2	- Mengalahkan 5 Mandrago-Witch
3	- Mengalahkan 3 Mandrago-Witch - Mengalahkan 3 Ghoulfly
4	- Mengalahkan 7 Mandrago-Witch - Mengalahkan 4 Ghoulfly
5	- Mengalahkan 5 Mandrago-Witch
6	- Mengalahkan 5 Mandrago-Witch - Mengalahkan 5 Ghoulfly
7	- Mengalahkan 7 Mandrago-Witch - Mengalahkan 4 Ghoulfly - Mengalahkan 1 PhantomReader
8	- Mengalahkan 10 Mandrago-Witch - Mengalahkan 10 Ghoulfly - Mengalahkan 1 PhantomReader
9	- Mengalahkan 15 Mandrago-Witch
10	- Mengalahkan 10 Mandrago-Witch - Mengalahkan 10 Ghoulfly
11	- Mengalahkan 10 Ghoulfly - Mengalahkan 4 PhantomReader - Mengalahkan 1 Corrupted FangJoker
12	- Mengalahkan 6 PhantomReader - Mengalahkan 3 Corrupted FangJoker

Ketika pemain memasuki *dungeon*, pemain dapat berjalan menyelusuri area yang disediakan. Pemain dapat menggerakkan karakter dengan menggunakan tombol arah. Selama menyelusuri *dungeon* pemain akan bertemu dengan musuh atau monster yang akan

dilawan. Musuh yang berada di *dungeon* di set menjadi tidak terlihat, sehingga pemain tidak akan menyadari jika disekitarnya ada musuh dan ketika pemain bersentuhan dengan musuh yang tak terlihat tadi, pemain akan masuk dalam *battle mode*. Penggambaran alur bertemu musuh akan dipresentasikan dalam *FSM Explore Dungeon* pada Gambar 3.1. Pemain dapat betemu 4 jenis musuh yang bisa dihadapi selama berada di *dungeon*. Namun, terdapat rasio kesempatan bertemu musuh pada setiap *stage*. Rasio untuk setiap *stage*-nya dijabarkan pada Tabel 3.4



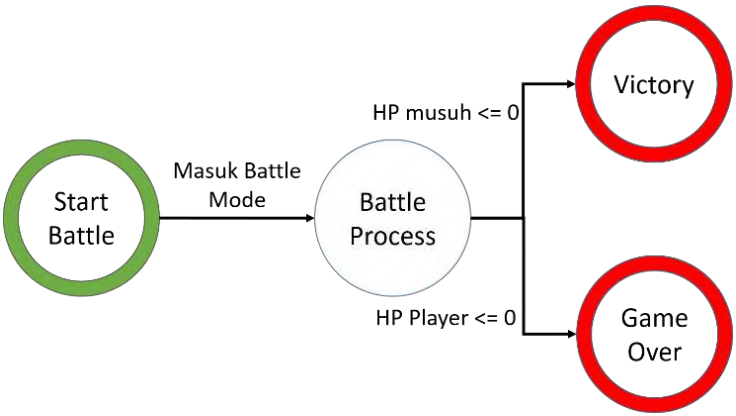
Gambar 3.1 *FSM Set Monster*

Tabel 3.4 *Rate Chance* per *Stage*

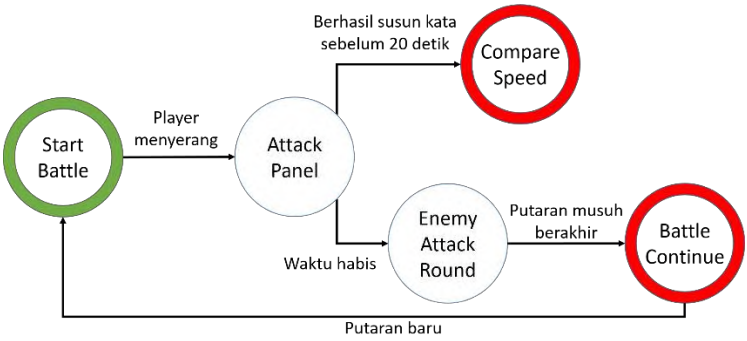
Stage	***Rate Chance***
1/Tutorial	- Mandrago-Witch 100%
1 – 4	- Mandrago-Witch 70% - Ghoulfly 30%
5 – 8	- Mandrago-Witch 45% - Ghoulfly 35% - PhantomReader 20%
9 - 12	- Mandrago-Witch 40% - Ghoulfly 30% - PhantomReader 20% - Corrupted FangJoker 10%

Pada *battle mode*, pemain dapat mengalahkan musuh dengan strategi yang dimiliki masing-masing. Pemain dapat memilih untuk menyerang musuh dengan serangan pedang, atau menyerang

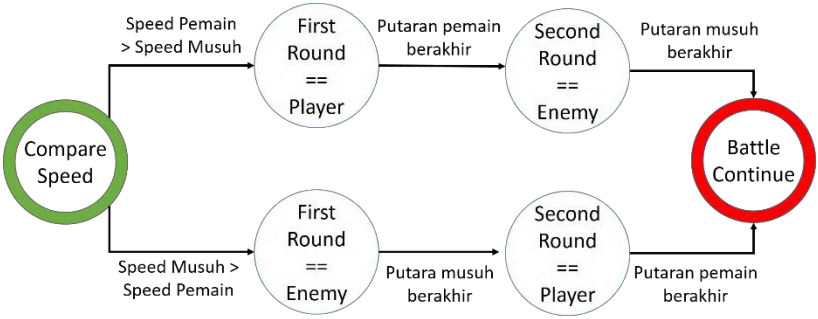
menggunakan *skill,* atau memulihkan diri dengan menggunakan *item*. Ketika pengguna menyerang musuh dengan serangan pedang, pemain akan dihadapkan dengan kumpulan huruf acak. Pemain harus menyusun huruf acak tersebut menjadi sebuah kata berbahasa Inggris yang benar. Terdapat batasan waktu dalam menyusun kata, yakni 20 detik. Jika pemain berhasil menyusun kata dengan benar, maka pemain akan menyerang musuh. Panjang kata yang disusun memengaruhi besar kecilnya nilai serangan. Perhitungan besar kecil nilai serangan akan dibahas pada alur memperbarui *skill*. Jika pemain salah dalam menyusun kata, maka pemain tidak dapat menyerang musuh, begitu pula jika pemain tidak mampu menyusun kata dalam waktu yang disediakan. Untuk musuh yang terkena efek *stun*, maka musuh juga tidak dapat menyerang. Dalam pertarungan, kecepatan tiap karakter sangat menentukan. Jika pemain atau musuh memiliki kecepatan yang lebih tinggi dari yang lain, maka pemilik kecepatan tertinggi akan mendapat giliran menyerang lebih dahulu. Pertarungan akan berakhir jika salah satu dari musuh ataupun pemain dengan *Health Point (HP)* kurang dari 0. Alur menyerang ini dijelaskan pada Gambar 3.2 sampai Gambar 3.6



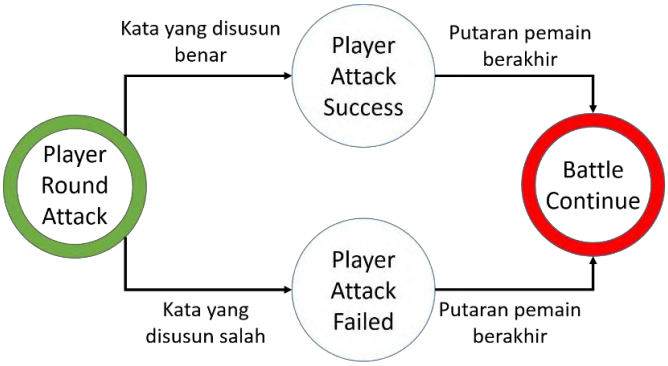
Gambar 3.2 *FSM* Menang Kalah



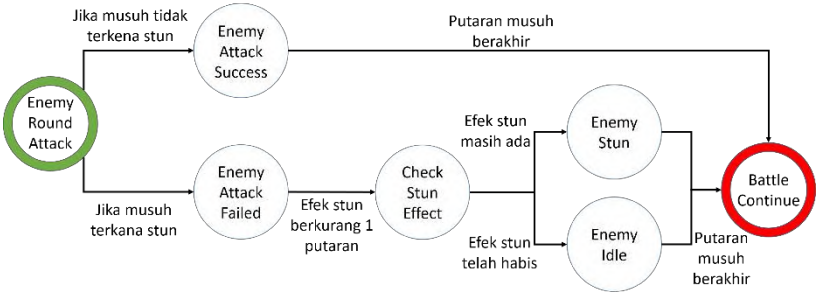
Gambar 3.3 *FSM* Durasi *Attack*



Gambar 3.4 *FSM Compare Speed*

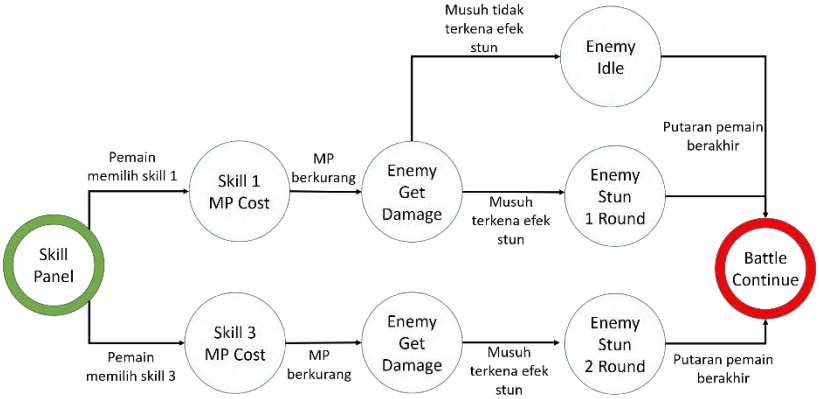


Gambar 3.5 *FSM Player Attack*



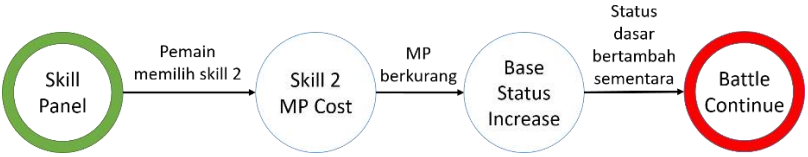
Gambar 3.6 *FSM Enemy Attack*

Dalam *panel skill*, pemain mendapat 3 pilihan *skill*. *Skill* 1 dan 3 adalah menggunakan *skill* yang menarget musuh. Musuh akan menerima serangan spesial dan mendapat kemungkinan tidak dapat bergerak. Untuk *skill* 1 musuh tidak dapat bergerak selama 1 putaran dengan kemungkinan 50%, sementara untuk *skill* 3, musuh tidak dapat bergerak selama 2 putaran dengan kemungkinan 100%. Setiap menggunakan *skill* akan mengurangi *MP*. *Skill* yang dimiliki pemain tergantung oleh elemen yang dipilih pemain pada awal cerita.. Sama dengan menyerang menggunakan pedang, kecepatan tiap karakter memengaruhi giliran menyerang Alur menyerang menggunakan *skill* 1 atau 3 dijelaskan pada Gambar 3.7.



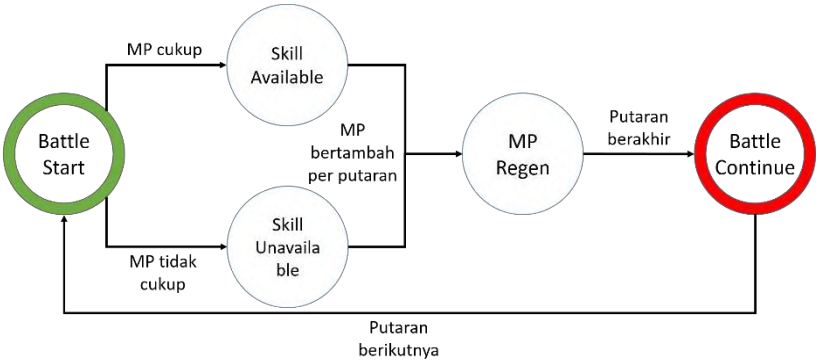
Gambar 3.7 *FSM* Penggunaan *Skill* 1 dan *Skill* 3

Skill 2 merupakan *skill* yang menarget pemain, sehingga ketika *skill* ini aktif, maka status pemain akan meningkat selama beberapa putaran.. Alur penggunaan *skill* 2 dijelaskan pada Gambar 3.8.



Gambar 3.8 *FSM* Penggunaan *Skill* 2

Untuk setiap putaran, *MP* pemain akan bertambah sebesar 5 poin. Alur penambahan *MP* dan mengecek ketersediaan *skill* dijelaskan pada Gambar 3.9.



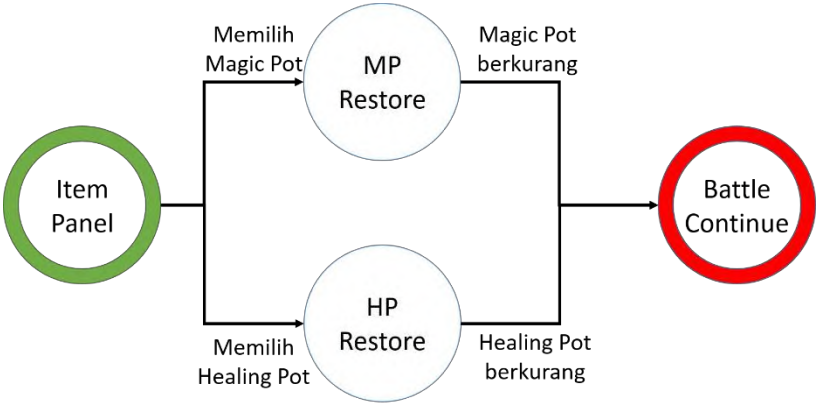
Gambar 3.9 *FSM MP Regen*

Pemain juga dapat menggunakan *item* dalam bertahan menghadapi serangan musuh. *Item* yang dapat digunakan ada empat, tergantung apakah pemain memiliki *item*-nya atau tidak. Terdapat 2 jenis *item*, satu untuk memulihkan *HP*, dan yang lain untuk memulihkan *MP*. Status tiap item ditunjukkan pada Tabel 3.5. Tidak sama dengan menyerang musuh, dalam menggunakan *item* kecepatan tidak akan memengaruhi giliran. Pemain yang menggunakan *item*

akan selalu berada pada giliran pertama, setelah itu musuh baru akan menyerang. Alur menggunakan *item* ini dijelaskan pada Gambar 3.10.

Tabel 3.5 Status *Item*

Nama *Item*	Status *Item*
Healing Potion	Memulihkan 150 *HP*
Super Healing Potion	Memulihkan 300 *HP*
Magic Potion	Memulihkan 50 *MP*
Super Magic Potion	Memulihkan 100 *MP*



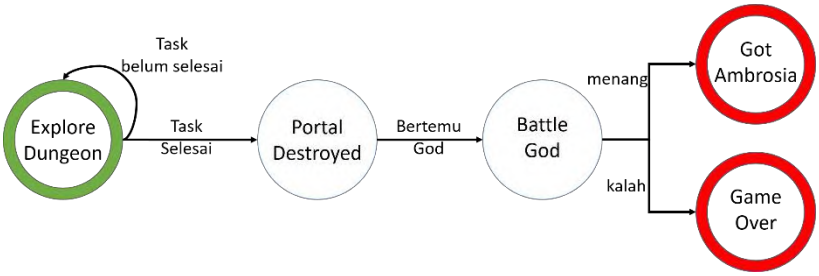
Gambar 3.10 *FSM Use Item*

Setelah berhasil mengalahkan musuh, pemain akan mendapat *experience (EXP)* dan gold. Jika *EXP* memenuhi, maka level pemain akan naik secara otomatis. Untuk mengetahui *EXP* yang dibutuhkan untuk naik level serta *EXP* dan *gold* yang diperoleh setiap mengalahkan musuh, akan dijabarkan pada alur status pemain dan alur status monster. Setelah mendapat *EXP* dan *gold*, pemain akan kembali ke dalam *dungeon*. Pemain harus menyelesaikan *task list* yang diberikan agar dapat menghancurkan penghalang atau portal menuju *boss stage*. Jika pemain mengalahkan musuh yang tidak ada pada *task list*, tidak akan merubah apa-apa. Ketika *task* yang diberikan sudah diselesaikan, pemain harus mencari dimana *God* atau *boss*

stage berada. Dalam melawan *God*, mekanisme pertempuran sama saja dengan melawan musuh biasa. Jika berhasil mengalahkan *God*, maka *dungeon stage* tersebut berhasil ditaklukkan dan pemain mendapat *Ambrosia*. Dan jika gagal, maka pemain akan kembali ke *main menu* dan tidak mendapat apa-apa. Untuk *reward* yang diperoleh setelah mendapat *Ambrosia*, dijabarkan dalam Tabel 3.6. Sedangkan alur melawan *God* direpresentasikan dalam Gambar 3.11.

Tabel 3.6 *Reward List*

Stage	***Reward List***
1/Tutorial	Membuka pilihan elemen (*reward* pertama) Mendapat 200 *gold* (*reward* kedua dan seterusnya)
2	Membuka *skill* pertama (*reward* pertama) Mendapat 200 *gold* (*reward* kedua dan seterusnya)
3	Mendapat 200 *gold*
4	Membuka *skill* kedua (*reward* pertama) Mendapat 300 *gold* (*reward* kedua dan seterusnya)
5	Mendapat 400 *gold*
6	Mendapat 500 *gold*
7	Mendapat 600 *gold*
8	Membuka *skill* ketiga (*reward* pertama) Mendapat 700 *gold* (*reward* kedua dan seterusnya)
9	Mendapat 800 *gold*
10	Mendapat 1000 *gold*
11	Mendapat 1200 *gold*
12	Mendapat 1500 *gold*



Gambar 3.11 *FSM* Melawan *Dewa*

Setiap musuh yang dilawan memiliki karakteristik tersendiri. Satu musuh dengan status standar, musuh dengan kecepatan yang tinggi tetapi memiliki serangan dan ketahanan yang rendah, musuh dengan ketahanan kuat tapi kecepatan rendah, dan musuh dengan status diatas musuh yang lain, sehingga total ada 4 jenis musuh yang bisa dilawan. Status dasar dari tiap musuh akan dijabarkan dalam Tabel 3.7. Agar tidak membosankan dalam melawan musuh, maka status setiap musuh akan berubah-ubah berdasarkan *stage* yang dimainkan pemain. Hal ini juga berlaku pada bos musuh, hanya saja status untuk boss berubah berdasarkan *dungeon map*. Perhitungan status musuh dilikukan secara otomatis dengan rumus yang ditampilkan pada Persamaan 3.1 untuk status musuh biasa dan pada Persamaan 3.2 untuk bos musuh.

Tabel 3.7 Status Dasar Musuh

Monster Name	*Atk*	*S Atk*	*Spd*	*HP*	*EXP*	*Gold*
Mandrago-Witch	10	17	10	50	12	10
Ghoulfly	10	20	20	40	10	10
PhantomReader	13	20	5	200	30	50
Corrupted FangJoker	30	75	25	300	100	200

$$NewStat = BaseStat + (BaseStat \times CurrentStage \times 10\%)$$

Persamaan 3.1 Rumus Perhitungan Status Dinamis

$$NewBossStat = NewStat + (NewStat \times CurrentMap \times 20\%)$$

Persamaan 3.2 Rumus Perhitungan Status Boss

Dalam permainan ini, pemain dapat melihat status karakter mereka. Untuk melihat status karakter, pemain perlu memasuki *panel Arsenal*. Pada *panel* tersebut, selain melihat status karakter, pemain juga dapat melihat keterangan alat tempur yang digunakan, jumlah *item* yang dimiliki, serta pilihan untuk meningkatkan *skill*. Untuk melihat keterangan alat tempur yang digunakan, pemain cukup menekan alat tempur tersebut. Hal ini juga berlaku untuk melihat keterangan *item*, pemain cukup menekan *item* yang dipilih. Namun, pemain terlebih dahulu harus berada pada *panel item*.

Status karakter pemain bergantung pada level dan elemen yang pemain pilih. Pengaturan *EXP* yang dibutuhkan untuk menaikkan level dilakukan secara otomatis dengan menggunakan rumus yang dijabarkan pada Persamaan 3.3. Pemain dapat menaikkan levelnya hingga level 20. Untuk setiap naik level, status karakter naik berdasarkan elemen yang dipilih. Jika pemain memilih elemen api, maka status yang dominan adalah *attack* dan *HP*, sedangkan jika pemain memilih elemen listrik, maka status yang dominan adalah *speed* dan *MP*. Representasi penambahan status seiap elemen terdapat pada Tabel 3.8, sedangkan nilai status karakter per levelnya terdapat pada Tabel 3.9.

$$EXP = \frac{(CurrentLevel + 1)^3 \times \big(100 - (CurrentLevel + 1)\big) - CurrentLevel^3 \times (100 - CurrentLevel)}{50}$$

Persamaan 3.3 Rumus Perhitungan *EXP*

Tabel 3.8 Penambah Status per Elemen

Atribut Status	Elemen Listrik	Elemen Api	Status/Level
Attack	+2	+3	1
Speed	+3	+2	1
HP	+10	+15	1
MP	+10	+5	5

Tabel 3.9 Status Karakter per Level

Lvl	**Elemen Listrik**				**Elemen Api**				*Next EXP*
	Atk	***Spd***	***HP***	***MP***	***Atk***	***Spd***	***HP***	***MP***	
1	10	10	100	50	10	10	100	50	14
2	12	13	110	50	13	12	115	50	37
3	14	16	120	50	16	14	130	50	71
4	16	19	130	50	19	16	145	50	115
5	18	22	140	60	22	18	160	55	169
6	20	25	150	60	25	20	175	55	232
7	22	28	160	60	28	22	190	55	304
8	24	31	170	60	31	24	205	55	385
9	26	34	180	60	34	26	220	55	473
10	28	37	190	70	37	28	235	60	569
11	30	40	200	70	40	30	250	60	672
12	32	43	210	70	43	32	265	60	782
13	34	46	220	70	46	34	280	60	897
14	36	49	230	70	49	36	295	60	1018
15	38	52	240	80	52	38	310	65	1144
16	40	55	250	80	55	40	325	65	1274
17	42	58	260	80	58	42	340	65	1409
18	44	61	270	80	61	44	355	65	1547
19	46	64	280	80	64	46	370	65	1688
20	48	67	290	90	67	48	385	70	1832

Pemain dapat meningkatkan *skill* yang mereka miliki. Untuk meningkatkan *skill*, pemain harus berada pada *skill panel* yang terletak di dalam *arsenal panel*. Setiap *skill* dapat ditingkatkan sebanyak tiga kali. Elemen yang dipilih juga memengaruhi *skill* yang dimiliki oleh pemain, kecuali *skill* pedang. Untuk pemain yang memilih elemen listrik maka karakteristik *skill* yang dimiliki adalah *damage* kecil dan konsumsi *mana* kecil. Sedangkan untuk elemen api, karakteristik *skill*-nya adalah *damage* besar, namun konsumsi *mana* cukup banyak. Pemain dapat memeroleh *skill* dengan menaklukkan *dungeon*. Bila pemain belum memiliki *skill* yang diperoleh dari

dungeon, maka pemain tidak dapat meningkatkannya. Hal yang dibutuhkan dalam meningkatkan *skill* adalah *gold*, jika *gold* yang dimiliki pemain memadai, pemain dapat meningkatkannya. Alur meningkatkan *skill* dijabarkan pada Gambar 3.12, sedangkan status setiap *skill* ditunjukkan dalam Tabel 3.10.

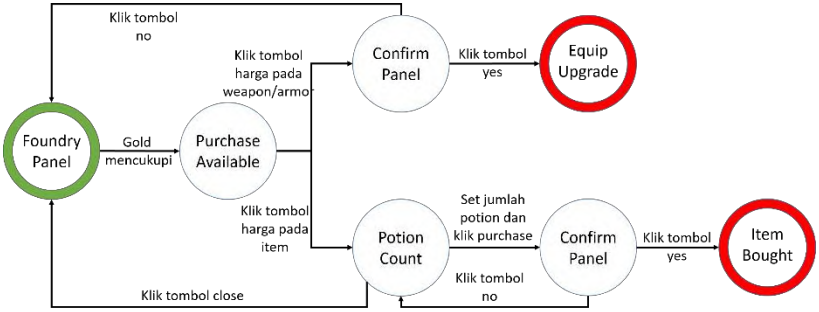
Gambar 3.12 *FSM* Meningkatkan *Skill*

Tabel 3.10 Status per *Skill*

Skill Name	**Level 0**	**Level 1**	**Level 2**	**Level 3**	***Cost***
Slash Strike	+10%DMG	+20%DMG Prc 500G	+30%DMG Prc 750G	+40%DMG Prc 1000G	-
Bolt Strike (elemen listrik)	150%DMG *Stun* 50% 1 *Turn*	165%DMG Prc 750G	180%DMG Prc 1250G	195%DMG Prc 1750G	20MP
Eerie Impulse (elemen listrik)	+10%ATK +20%SPD *Buff* 2 *Turn*	+20%ATK +40%SPD Prc 1250G	+30%ATK +60%ATK Prc 1750G	+40%ATK +80%SPD Prc 2250G	35MP
Zap Cannon (elemen listrik)	240%DMG *Stun* 100% 2 *Turn*	280%DMG Prc 2000G	320%DMG Prc 2750G	360%DMG Prc 3500G	55MP
Flare Blitz (elemen api)	150%DMG *Panic* 50% 1 *Turn*	170%DMG Prc 750G	190%DMG Prc 1250G	210%DMG Prc 1750G	30MP
Will - o - Wisp (elemen api)	+20%ATK +10%SPD *Buff* 2 *Turn*	+40%ATK +20%SPD Prc 1250G	+60%ATK +30%ATK Prc 1750G	+80%ATK +40%SPD Prc 2250G	40MP

Blaze Burn (elemen api)	250%DMG *Panic*100% 2 *Turn*	300%DMG Prc 2000G	350%DMG Prc 2750G	400%DMG Prc 3500G	60MP

Beralih ke *foundry panel*, dalam *panel* ini pemain dapat membeli *item* atau meningkatkan alat tempur mereka. Syarat yang diperlukan dalam membeli atau meningkatkan alat tempur adalah memilik *gold* yang mencukupi. Pemain dapat meningkatkan alat tempur sebanyak lima kali. Sedangkan untuk membeli *item*, pemain dapat membeli lebih dari satu *item* yang sama dalam satu proses. Setiap senjata atau armor yang ditingkatkan akan mengalami perubahan status. Status setiap level dari alam tempur ditampilkan dalam Tabel 3.11. Untuk alur dari membeli atau meningkatkan barang direpresentasikan dalam Gambar 3.13.



Gambar 3.13 *FSM* Meningkatkan atau Membeli Barang

Tabel 3.11 Status Alat Tempur per Level

Level	***Status Weapon***	***Status* Armor**
0	ATK +3	HP +5
1	ATK +10 *Price* 500	HP +10 *Price* 350
2	ATK +32 *Price* 1500	HP +60 *Price* 1000
3	ATK +66 *Price* 2500	HP +120 *Price* 1850

4	ATK +112 *Price* 3500	HP +240 *Price* 2600
5	ATK +170 *Price* 4500	HP +360 *Price* 3350

3.2.2 Perancangan Antarmuka Permainan

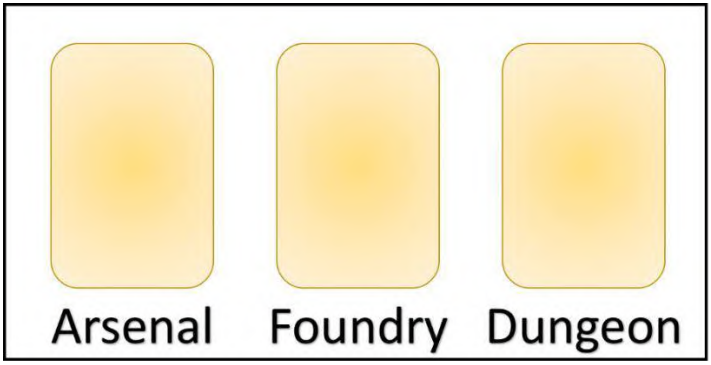
Perancangan antarmuka digunakan untuk memberikan gambaran terhadap permainan yang akan dikembangkan. Dengan adanya perancangan antarmuka, maka akan memudahkan dalam memahami permainan ini. Perancangan antarmuka ini dirancang menggunakan Microsoft Power Point 2013 dan Paint serta beberapa sketsa gambar dari Google.

3.2.2.1 Rancangan *Welcome Screen*

Gambar 3.14 Tampilan *Welcome Screen*

Antarmuka ini merupakan antarmuka pertama yang akan dilihat oleh pemain. Tombol yang terdapat antarmuka ini adalah tombol *start* untuk masuk kedalam permainan, tombol *sound* untuk mengatur suara dan musik, serta tombol *credit* untuk menampilkan informsi yang berkontribusi dalam pembuatan permainan ini.

3.2.2.2 Rancangan *Main Menu*



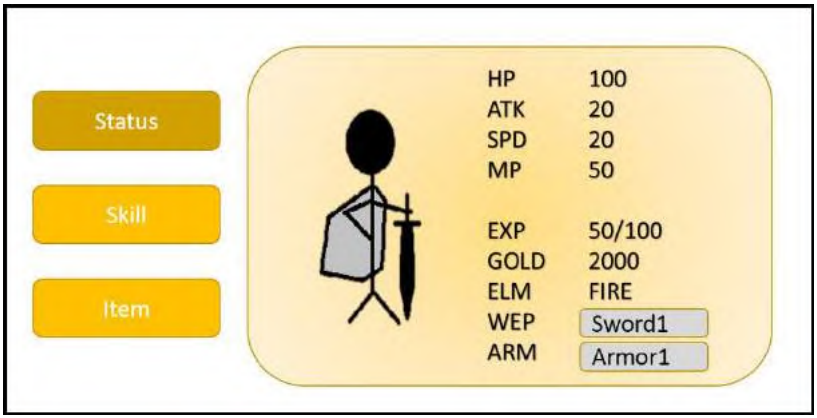
Gambar 3.15 Tampilan *Main Menu*

Pemain akan melihat antarmuka *main menu* ketika pemain menakan tombol *start* yang terdapat pada tampilan sebelumnya. Dalam antarmuka ini, pemain akan melihat 3 tombol, yakni tombol *arsenal* untuk masuk ke menu *arsenal* dimana pemain dapat melihat statusnya, tombol *foundry* untuk masuk menu *foundry* dimana pemain dapat meningkatkan peralatannya, dan tombol *dungeon* untuk masuk ke dalam *dungeon map* dan memilih *stage*/.

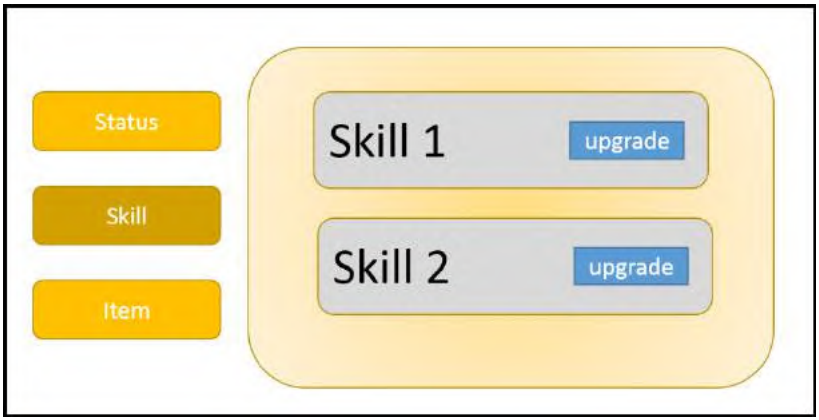
3.2.2.3 Rancangan *Arsenal Menu*

Pada menu ini, pemain dapat melihat status dari karakter yang dimainkan. Status yang dapat dilihat pemain adalah *HP, attack, speed, MP, EXP, gold,* elemen yang dipilih, serta senjata dan armor yang digunakan. Selain melihat status, pemain dapat meningkatkan *skill* yang dimiliki serta melihat *item* yang dimiliki. Dalam *skill panel*, pemain akan melihat *skill* yang dimiliki dan tombol *upgrade* untuk meningkatkannya. *Skill* yang tidak dimiliki tidak akan ditampilkan pada layar pemain. Sedangkan pada *item* panel, pemain akan melihat *item* yang tersedia pada permainan beserta jumlah yang dimiliki pemain. Secara *default*, ketika pemain memasuki *arsenal menu*, maka pemain akan melihat panel status terlebih dahulu. Untuk melihat

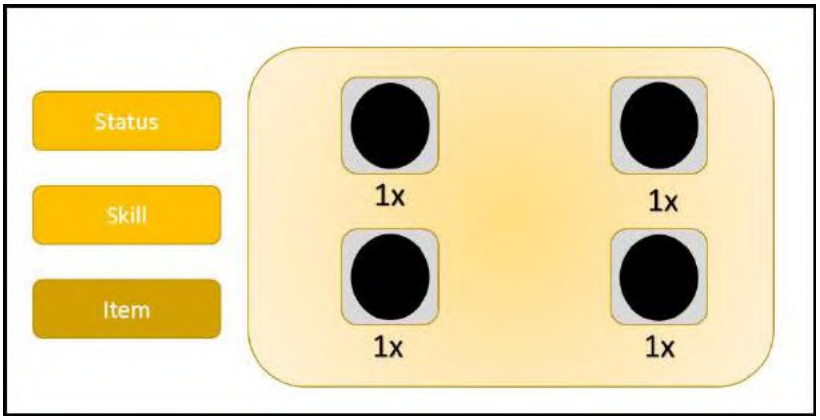
rancangan panel status terdapat pada Gambar 3.16, untuk melihat rancangan panel *skill* terdapat pada Gambar 3.17, dan untuk melihat rancangan panel *item* terdapat pada Gambar 3.18.



Gambar 3.16 Tampilan Status Panel

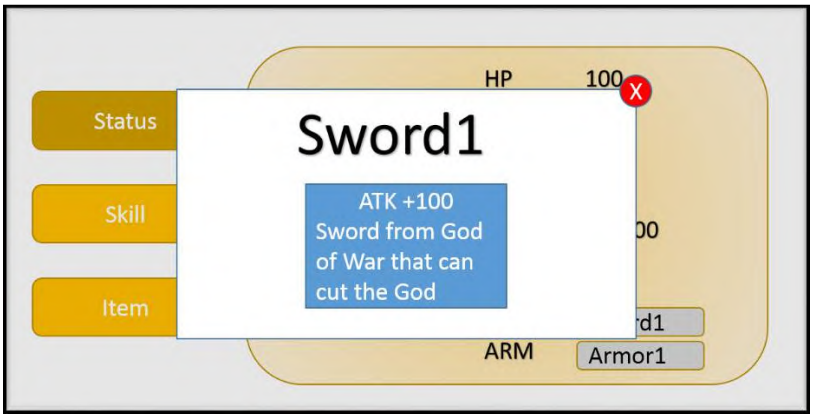


Gambar 3.17 Tampilan *Skill* Panel



Gambar 3.18 Tampilan *Item* Panel

Dalam menu ini, pemain juga dapat melihat *preview* atau informasi singkat terhadap senjata, armor, dan item yang dimiliki. Pemain cukup menekan tombol pada senjata atau pada armor atau pada *item* untuk melihatnya. *Preview* panel ditampilkana pada Gambar 3.19.



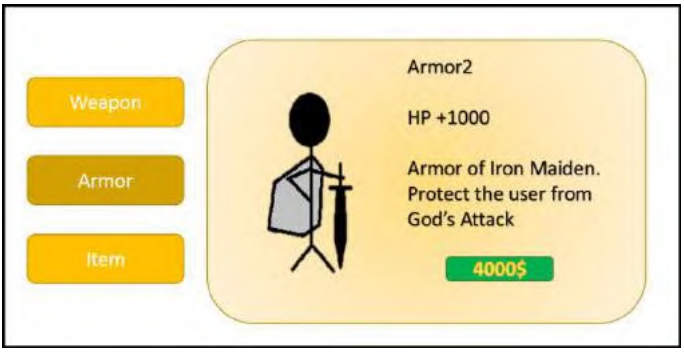
Gambar 3.19 Tampilan *Preview* Panel

3.2.2.4 Rancangan *Foundry Menu*

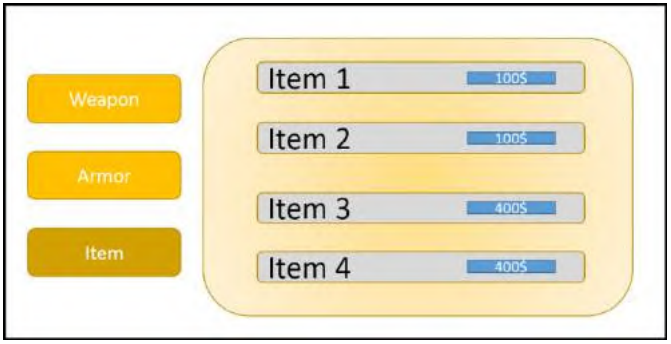
Pada menu ini, pemain dapat meningkatkan senjata dan armor serta membeli beberapa *tiem*. Pada bagian kiri terdapat 3 tombol yang digunakan untuk berpindah panel, yakni tombol *weapon*, armor, dan *item*. Secara *default*, pemain akan berada pada *weapon* panel. Pada *weapon* panel, pemain dapat melihat senjata pada bagian kiri panel kuning yang akan berubah jika diperbarui. Kemudian ditampilkan pula nama senjata yang akan diperbarui beserta keterangan dan statusnya. Pada bagian bawah panel kuning, terdapat tombol untuk meningkatkan senjata, dimana teks yang ditampilkan adalah harga untuk meningkatkan senjata. Sama halnya dengan *weapon* panel, pada armor panel memiliki tampilan yang sama, gambar dan keterangan beserta harganya menyesuaikan dengan armor yang akan diperbarui.

Untuk *item* panel terdapat beberapa perbedaan dibanding panel sebelumnya. Akan terdapat 4 panel untuk setiap *tiem*, didalamnya terdapat nama dari *item* tersebut dan tombol harga yang menyesuaikan dengan *item*-nya. Ketika pemain menekan tombol harga, akan muncul *pop up* yang menampilkan pilihan jumlah *item* yang akan dibeli. Rancangan meningkatkan senjata dapat dilihat pada Gambar 3.20, untuk melihat rancangan meningkatkan armor terdapat pada Gambar 3.21, dan untuk melihat rancangan membeli *item* terdapat pada Gambar 3.22 serta untuk rancangan *pop up* jumlah item ditampilkan pada Gambar 3.23.

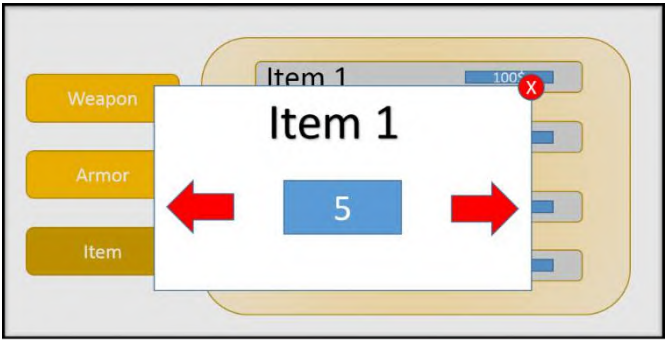
Gambar 3.20 Tampilan *Upgrade Weapon*



Gambar 3.21 Tampilan *Upgrade* Armor

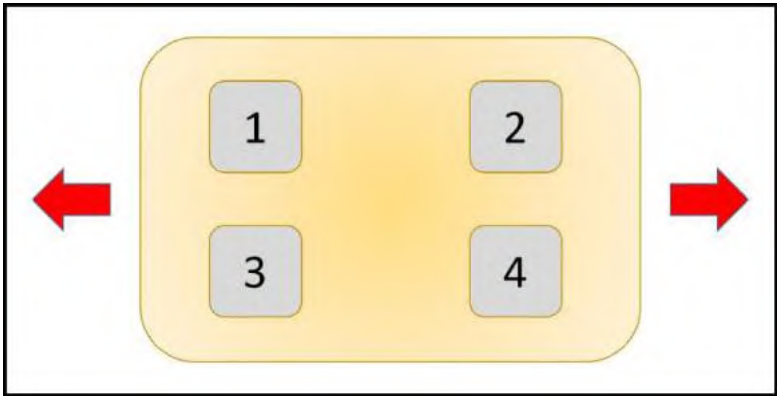


Gambar 3.22 Tampilan *Buy Item*



Gambar 3.23 Tampilan *Pop Up Item Count*

3.2.2.5 Rancangan *Dungeon Menu*



Gambar 3.24 Tampilan *Dungeon* Panel

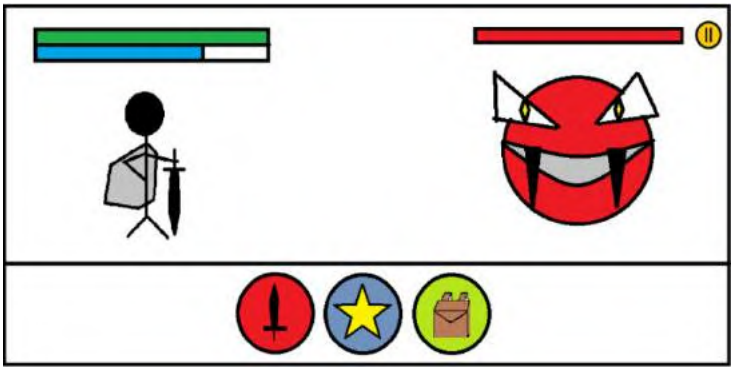
Gambar 3.25 Tampilan *Task List*

Pada menu ini, pemain akan memilih *dungeon* untuk dimainkan. Tombol yang ditampilkan pada pengguna berupa pemilihan *dungeon* dan tombol *next/prev* untuk mengganti *dungeon map*. Dan jika pengguna memilih *dungeon* tersebut, akan ditampilkan *pop up task list* yang berisi hal apa saja yang harus dilakukan pemain selama berada di *dungeon*. Untuk melihat rancangan pemilihan *dungeon* dapat dilihat pada Gambar 3.24, sedangkan untuk melihat *task list* setiap *dungeon* dapat dilihat pada Gambar 3.25.

Gambar 3.26 Tampilan *Explore Dungeon*

Pada Gambar 3.26, rancangan tampilan untuk *explore dungeon* diambil dari salah satu potongan gambar pada *game* Pokemon. Pada antarmuka tersebut, terdapat dua tombol, yakni tombol *analog* untuk menggerakkan karakter dan tombol *pause* untuk menjeda permainan.

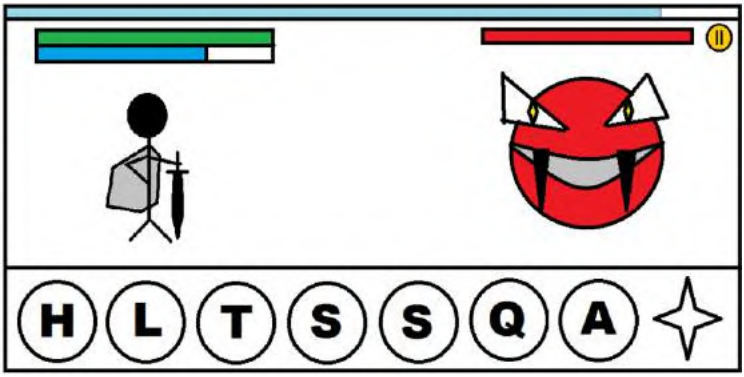
3.2.2.6 Rancangan *Battle Mode*



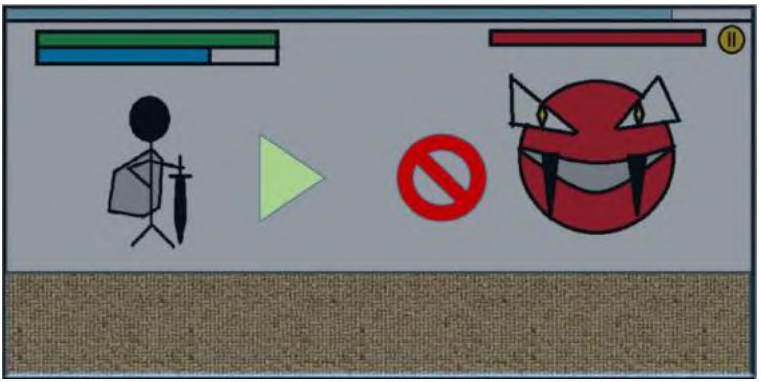
Gambar 3.27 Tampilan Awal *Battle Mode*

Gambar diatas merupakan tampilan awal dalam *battle mode*. Pemain akan melihat 2 karakter, dimana karakter pemain berada di sebelah kiri dan karakter musuh pada sebelah kanan. Diatas setiap karakter terdapat *health bar* yang menjadi indikator *HP* yang dimiliki. Untuk karakter pemain, terdapat indikator tambahan, yakni *mana bar* untuk mengetahui *MP* saat ini. Terdapat 3 tombol utama yang dapat pemain pilih yang terletak di bawah. Tombol pertama adalah untuk masuk ke dalam *attack* panel, tombol kedua untuk masuk ke dalam *skill* panel, dan tombol terakhir untuk masuk ke dalam *item* panel. Selain itu, juga terdapat tombol jeda pada pojok kanan atas.

Ketika pemain menekan tombol *attack*, maka beberapa tampilan akan berubah. Perbedaan tersebut antara lain pada sisi atas tampilan akan terdapat *time bar* yang menandakan waktu yang tersisa dalam menyusun kata. Pada bagian bawah, akan terdapat 8 tombol, dimana 7 tombol menampilkan huruf acak, dan tombol ke-delapan untuk *submit* kata yang telah disusun. Rancangan antarmuka *attack mode* ditampilkan pada Gambar 3.28. Jika pemain menekan tombol jeda, maka permainan dijeda dan akan terdapat panel hitam dibagian bawah agar pemain tidak dapat melihat huruf yang sedang ditampilkan. Pada panel jeda ini terdapat 2 tombol, yakni tombol *resume* dan tombol *exit* untuk keluar ke *main menu.* Rancangan antarmuka untuk jeda saat *attack mode* dapat dilihat pada Gambar 3.29.

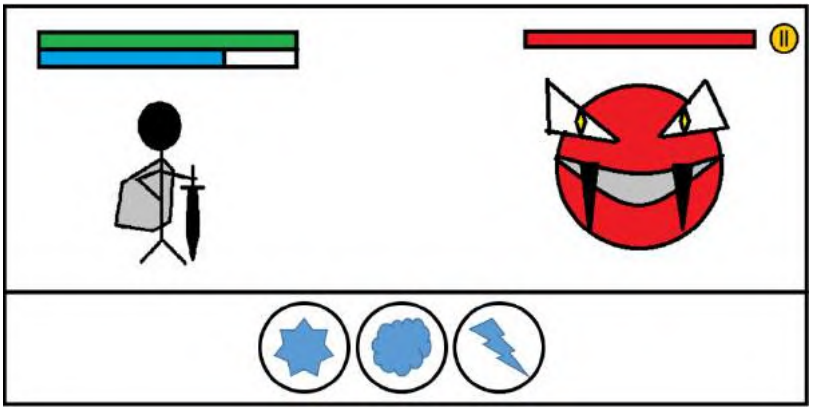


Gambar 3.28 Tampilan *Attack Mode*



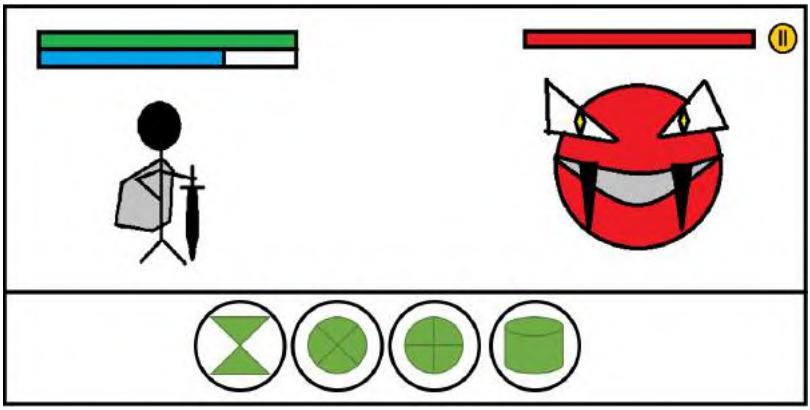
Gambar 3.29 Tampilan *Pause* saat *Attack Mode*

Tampilan ketika berada pada *skill mode* hampir sama dengan tampilan awal *battle mode*. Hanya saja, 3 tombol pada bagian bawah merupakan tombol untuk menggunakan *skill*. Tombol pertama untuk menggunakan *skill basic*, tombol kedua untuk menggunakan *skill buff*, dan tombol ketiga untuk menggunakan *skill ultimate*. Ikon tiap *skill* tidak akan sama antar elemen satu dengan elemen yang lain. Rancangan antarmuka *skill mode* ditampilkan pada Gambar 3.30.



Gambar 3.30 Tampilan *Skill Mode*

Untuk tampilan pada *item mode*, terdapat 4 tombol yang berbeda pada bagian bawah tampilan. Setiap tombol tersebut merupakan *item-item* yang dapat digunakan pemain untuk menambah *MP* ataupun *HP*. Rancangan antarmuka *item mode* ditampilkan pada Gambar 3.31.



Gambar 3.31 Tampilan *Item Mode*

3.2.3 Perancangan Algoritma Permainan

Tahap perancangan algoritma yang akan diterapkan dalam permainan dibagi menjadi, yakni merancang kamus referensi dan merancang pengacakan dan pengecekan kata.

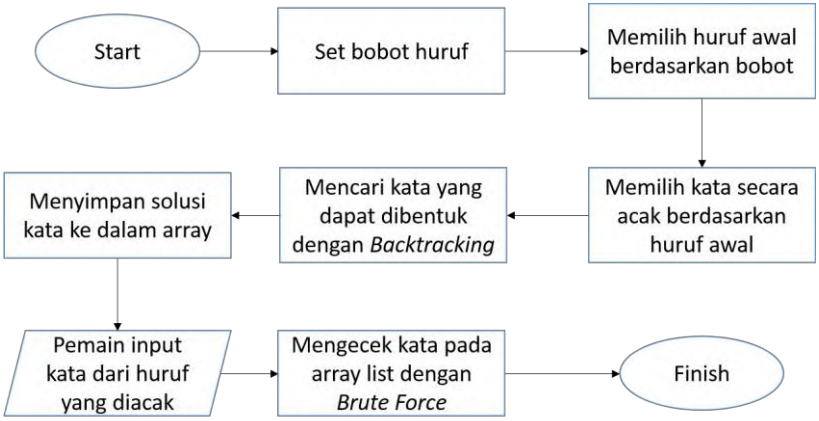
3.2.3.1 Perancangan Kamus Referensi

Kamus referensi yang digunakan dalam permainan ini menggunakan kumpulan kata bahasa inggris yang bersumber dari *Infochimps* [10]. Kata yang telah dikumpulkan oleh *Infochimps* berjumlah sebanyak 354986 kata. Dalam kata yang dikumpulkan tersebut tidak mengandung kata jamak. Kata-kata dalam kamus referensi mengandung beberapa singkatan dan nama yang familiar dalam bahasa inggris, serta beberapa kata lampau juga termasuk di dalamnya. Selanjutnya akan dilakukan proses filterisasi dimana hanya akan mengambil kata dengan panjang minimal 3 huruf dan maksimal

7 huruf yang digunakan dalam permainan, serta memisah kamus menjadi 26 bagian berdasarkan huruf awalnya.

3.2.3.2 Perancangan Pengacakan dan Pengecekan Kata

Fokus dalam pengerjaan Tugas Akhir ini terletak pada *gameplay* dimana diterapkannya Algoritma *Backtracking* dalam pengecekan kata. Alur dari perancangan pengacakan dan pengecekan kata direpresentasikan pada Gambar 3.32.



Gambar 3.32 *Flow Chart* Pengacakan dan Pengecekan Kata

Langkah awal yang digunakan adalah menentukan bobot untuk setiap huruf. Penentuan bobot setiap huruf didasarkan oleh poin setiap huruf pada permainan Scrabble [11]. Poin setiap huruf yang terdapat pada permainan Scrabble diasumsikan sebagai nilai frekuensi keluarnya suatu huruf, dimana jika poin huruf rendah maka huruf tersebut sering muncul begitu pula sebaliknya, jika poin tinggi maka huruf tersebut jarang keluar. Poin setiap huruf dapat dilihat pada Tabel 3.12. Setelah mendapat poin setiap huruf, kemudian dicari nilai maksimum dari semua huruf berdasarkan frekuensinya. Setelah itu dicari bobot setiap huruf dengan membagi frekuensi setiap huruf dengan nilai maksimum. Rumus untuk mencari frekuensi setiap huruf dapat dilihat pada Persamaan 3.4, sedangkan rumus untuk mencari

nilai maksimum terletak pada Persamaan 3.5 dan untuk rumus mencari bobot setiap huruf dapat dilihat pada Persamaan 3.6. Bobot setiap huruf ditampilkan pada Tabel 3.13.

Tabel 3.12 Poin per Huruf

Huruf	Bobot	Huruf	Bobot
A	1	N	2
B	4	O	1
C	4	P	4
D	2	Q	10
E	1	R	1
F	4	S	1
G	3	T	1
H	3	U	2
I	1	V	5
J	10	W	4
K	5	X	8
L	2	Y	3
M	4	Z	10

$$\boldsymbol{FrekuensiHurufi = \frac{10}{PoinHuruf_i} \times 12}$$

Persamaan 3.4 Rumus Mencari Frekuensi Setiap Huruf

$$\boldsymbol{MaxNilai = \sum_{i=1}^{26} FrekuensiHuruf_i}$$

Persamaan 3.5 Rumus Mencari Nilai Maksimum

$$BobotHuruf_i = \frac{FrekuensiHuruf_i}{MaxNilai} \times 100\%$$

Persamaan 3.6 Rumus Mencari Bobot Setiap Huruf

Langkah berikutnya adalah menentukan huruf awal menggunakan Algoritma *Roulette Wheel Selection*, dimana akan dilakukan pengacakan berdasarkan bobot setiap huruf yang telah

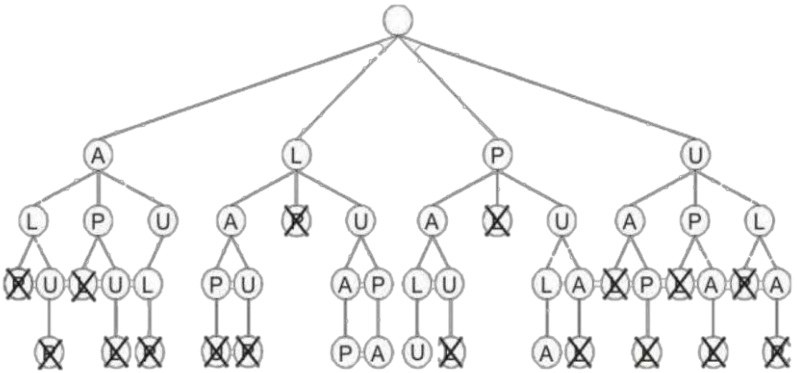
diperoleh. Pada proses ini huruf dengan bobot yang besar memiliki peluang untuk terpilih lebih tinggi dibanding huruf dengan bobot kecil. Setelah mendapat huruf awal, langkah berikutnya memilih kata secara acak berdasarkan huruf awal yang terpilih. *Input* dari proses ini adalah huruf awal yang diperoleh pada proses sebelumnya, kemudian akan dipilih kamus referensi yang mewakiliki huruf awal tersebut. Lalu dilakukan pemilihan kata secara acak dengan rentang banyaknya kata pada kamus refensi yang telah terpilih. *Output* dari proses ini adalah terpilihnya sebuah kata.

Tabel 3.13 Bobot per Huruf

Huruf	Bobot	Huruf	Bobot
A	8%	N	4%
B	2%	O	8%
C	2%	P	2%
D	4%	Q	1%
E	8%	R	8%
F	2%	S	8%
G	3%	T	8%
H	3%	U	4%
I	8%	V	1%
J	1%	W	2%
K	2%	X	1%
L	4%	Y	3%
M	2%	Z	1%

Proses berikutnya adalah mencari solusi kata yang dapat dibentuk dengan Algoritma *Backtracking*. *Input* dari proses ini adalah kata yang telah terpilih pada proses sebelumnya. Sebagai contoh, kata yang terpilih adalah "LUPA". Dari kata tersebut, akan dihidupkan simpul untuk setiap huruf sesuai dengan alphabet, dimulai dengan huruf "A". Dari simpul "A" kemudian dihidupkan simpul "L". Lalu dicek kedalam kamus kata dengan awalan *string* "AL". Jika ada, maka pencarian dilanjutkan dengan menghidupkan simpul "P" lalu dicek kembali kedalam kamus kata dengan awalan "ALP". Jika tidak ada,

maka simpul “P” diputus dan *backtrack* ke simpul “L”. Solusi ditemukan jika *string* yang terbentuk dari lintasan simpul sama dengan kata yang terdapat pada kamus referensi. Pencarian dilakukan hingga ditemukan solusi dan tidak dapat lagi terbentuk suatu lintasan. Contoh proses pencarian kata dapat dilihat pada Gambar 3.33. Proses selanjutnya adalah setiap solusi yang ditemukan akan dimasukkan kedalam *array list*.



Gambar 3.33 Contoh Pencarian Kata

Pada proses berikutnya, pemain akan memasukkan sebuah kata berdasarkan kumpulan huruf acak yang ditampilkan pada bagian *gameplay* hasil *output* dari proses pemilihan kata. Kata yang telah dimasukkan oleh pemain kemudian dicek dengan *array list* yang telah terbentuk menggunakan Algoritma *Brute Force*. Pengecekannya dilakukan dengan membandingkan *string*, bukan dengan membandingkan setiap karakter atau huruf.

(Halaman ini sengaja dikosongkan)

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi dari perancangan permainan. Di dalamnya mencakup proses penerapan dan pengimplementasian algoritma, dan antarmuka yang mengacu pada rancangan yang telah dibahas sebelumnya.

4.1 Lingkungan Implementasi

Lingkungan implementasi merupakan lingkungan dimana aplikasi akan dibangun. Lingkungan implementasi dibagi menjadi dua, yaitu lingkungan implementasi untuk pengembangan apilkasi yang ditunjukkan pada Tabel 4.1 dan lingkungan implementasi untuk *debugging* yang ditunjukkan pada Tabel 4.2.

Tabel 4.1 Lingkungan Implementasi Pengembang

Perangkat Keras	Tipe: MSI PE60-2QE Prosesor: Intel(R) Core(TM) i7-5700HQ CPU @ (8 Thread Core) 2.70GHz – 3.50GHz Memori: 8096 MB DDR3L GPU: Nvidia GTX 960M
Perangkat Lunak	Sistem Operasi: Windows 10 Pro 64-bit Perangkat Pengembang: Unity3D, CodeBlocks *Text Editor*: Sublime 3

Tabel 4.2 Lingkungan Implementasi *Debugging*

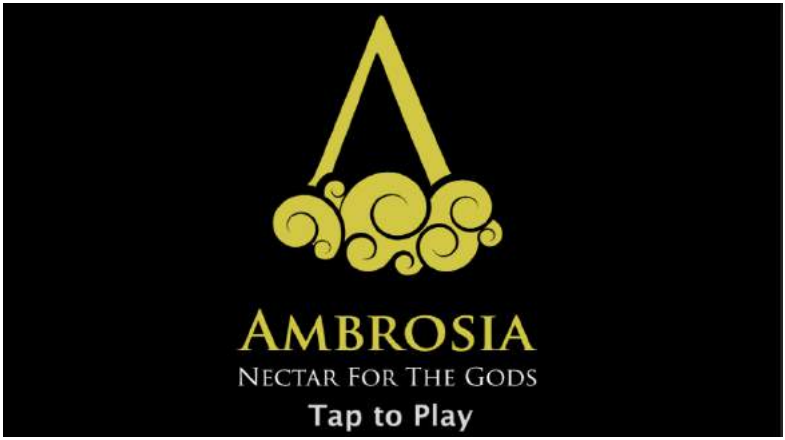
Perangkat Keras	Tipe: Lenovo Vibe X2 Prosesor: MediaTek MT6595M Memori: 2048MB RAM GPU: PowerVR G600
Perangkat Lunak	Sistem Operasi: Android Lollipop 5.0

4.2 Implementasi Antarmuka Permainan

Pada subbab ini, akan dijelaskan implementasi dari perancangan antarmuka yang terdapat pada bab 3. Terdapat 4 antarmuka utama dalam permainan ini antara lain antarmuka *welcome screen, main menu, dungeon,* dan *battle*.

4.2.1 Implementasi Antarmuka *Welcome Screen*

Tampilan pertama yang akan dilihat oleh pemain adalah antarmuka *welcome screen*. Pada antarmuka ini, pemain akan melihat logo dari *game Ambrosia*. Hanya ada satu tombol pada antarmuka ini, dan tombol tersebut terletak memenuhi semua tampilan. Tombol di set transparan, sehingga logo permainan tidak tertutupi. Ketika pemain menekan layar atau tombol tersebut, maka pemain akan masuk ke dalam antarmuka *main menu*. Antarmuka *welcome screen* dapat dilihat pada Gambar 4.1.



Gambar 4.1 Antarmuka *Welcome Screen*

4.2.2 Implementasi Antarmuka *Main Menu*

Antarmuka *main menu* merupakan antarmuka utama dalam permainan ini. Antarmuka *main menu* terbagi menjadi 4 panel, yakni *dungeon* panel, *arsenal* panel, *foundry* panel, dan

other panel. Pada sisi atas kanan, terdapat 4 tombol dimana setiap tombol mewakili satu panel. Sedangkan pada sisi kirinya, menampilkan nama dari karakter dalam *game* ini beserta levelnya, uang atau gold yang dimiliki, serta progres *EXP* untuk naik ke level selanjutnya. Tombol dan mini status tersebut merupakan *default* pada *main menu* dan digunakan pada semua panel.

4.2.2.1 Implementasi Antarmuka *Dungeon* Panel

Pada saat memasuki antarmuka *main menu*, pemain akan langsung berada pada *dungeon* panel sebagai *default* panel. Seperti yang dijelaskan pada subbab 4.2.2, pada *dungeon* panel akan memuat tampilan *default main menu*. Ketika berada pada panel ini, maka tombol *dungeon* pada sisi atas akan di nonaktifkan. Panel ini menampilkan *dungeon map*, serta *dungeon stage* yang dapat dipilih oleh pemain. Setiap *dungeon map* memiliki 4 stage. Pemain dapat mengganti *dungeon map* dengan menekan tombol *next/prev* yang berada pada sisi panel. Antarmuka panel ini dapat dilihat pada Gambar 4.2.

Gambar 4.2 Antarmuka *Dungeon* Panel

Setelah pemain memilih *stage* yang akan dimainkan, maka akan muncul *pop up* yang menampilkan *task list* yang harus

dilakukan dalam *dungeon*. Dalam *task list* panel, pemain dapat melihat nama *stage* yang akan dimainkan pada sisi atas panel. Sedangkan pada sisi tengah panel menjelaskan isi dari *task list*. Terdapat 2 tombol pada *pop up* ini, yakni tombol enter untuk memasuki *dungeon*, dan tombol close untuk menutup *pop up*. Antarmuka *pop up task list* dapat dilihat pada Gambar 4.3.

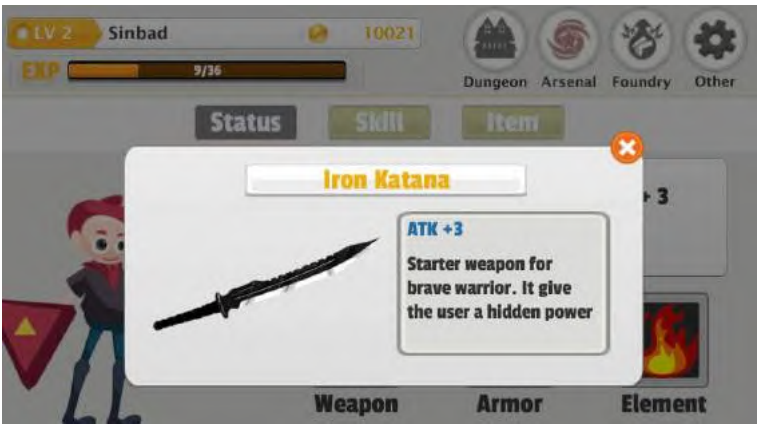
Gambar 4.3 Antarmuka *Pop Up Task List*

4.2.2.2 Implementasi Antarmuka *Arsenal* Panel

Pemain dapat masuk ke dalam *arsenal* panel setelah menekan tombol *arsenal* pada tombol utama. Dalam *arsenal panel*, terdapat tiga tombol pada sisi atas. Mulai dari sebelah kiri, yakni tombol status, *skill*, dan *item*. Tombol status akan dipilih secara *default* ketika pemain masuk ke dalam *arsenal* panel. Ketika tombol status dipilih, pemain dapat melihat status karakter yang ditampilkan pada dua panel sebelah gambar karakter. Status yang ditampilkan adalah *HP*, *MP*, *attack*, dan *speed*. Sementara pada bagian bawahnya, terdapat 3 panel, yakni panel senjata, armor dan elemen. Setiap panel pada bagian bawah dapat memunculkan *pop up preview* dari panel yang dipilih. Isi dari *pop up preview* yakni pada bagian atas terdapat nama dari panel yang dipilih, dan

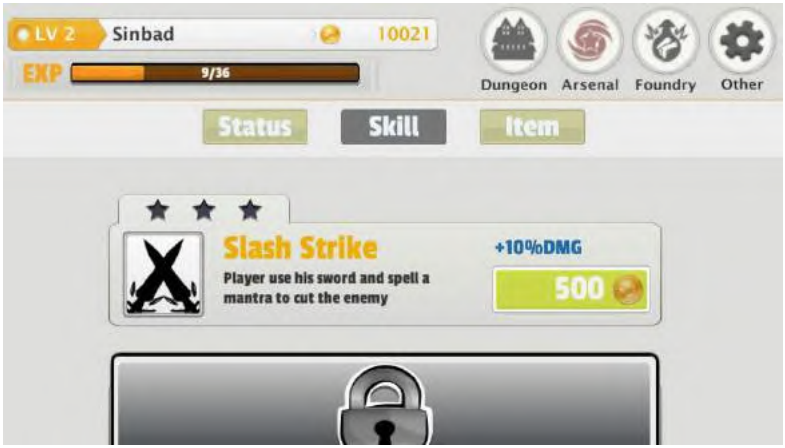
dibawahnya terdapat keterangan dari setiap panel beserta status apa saja yang mendapat tambahan dari alat tempur atau elemen tersebut. Tombol pada *pop up preview* hanya ada satu, yakni tombol *close*. Selama *pop up review* aktif, pemain tidak dapat mengakses tombol lainnya. Antarmuka status karakter dapat dilihat pada Gambar 4.4, sedangkan antarmuka *pop up preview* dapat dilihat pada Gambar 4.5.

Gambar 4.4 Antarmuka Status Karakter



Gambar 4.5 Antarmuka *Pop Up Preview*

Pemain dapat meningkatkan *skill* karakter dengan menekan tombol *skill* pada bagian atas *arsenal* panel. Ketika tombol *skill* dipilih, pemain dapat melihat *skill* yang dimiliki saat ini. Terdapat 4 *skill* yang bisa ditingkatkan. Pemain bisa menggulir ke atas untuk melihat *skill* yang tidak nampak di layar. *Skill* yang tidak dimiliki akan terkunci dan tidak dapat ditingkatkan. Untuk *skill* yang telah dimiliki, pemain dapat meningkatkan *skill* tersebut sebanyak tiga kali. Terdapat indikator berupa bintang pada bagian atas panel tiap *skill* yang menunjukkan berapa kali *skill* tersebut ditingkatkan. Untuk bagian dalam panel *skill* terdapat ikon dari masing-masing *skill*, tepat disebelahnya adalah nama dan keterangan *skill*, dan pada bagian kanan adalah status atau nilai serangan beserta tombol yang menampilkan harga dari *skill* tersebut. Tombol tersebut akan aktif jika uang atau *gold* yang dimiliki cukup untuk melakukan peningkatan *skill*. Antarmuka meningkatkan *skill* dapat dilihat pada Gambar 4.6.



Gambar 4.6 Antarmuka *Upgrade Skill*

Tombol ketiga yang dapat dipilih pada *arsenal* panel adalah tombol *item*. Setelah dipilih, *arsenal* panel akan menampilkan list *item* yang ada pada permainan beserta jumlah yang dimiliki oleh pemain pada bagian bawah setiap *item*. Setiap

item dapat menampilkan *pop up preview* ketika *item* tersebut ditekan. Konten dari *pop up item* sama dengan *pop up preview* pada bagian status. Untuk melihat antarmuka *item list* terdapat pada Gambar 4.7, sedangkan untuk *pop up item* terdapat pada Gambar 4.8.

Gambar 4.7 Antarmuka *Item List*

Gambar 4.8 Antarmuka *Pop Up Item*

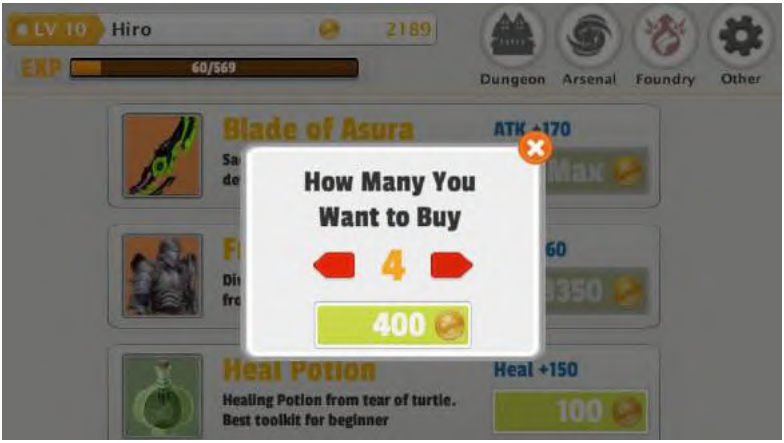
4.2.2.3 Implementasi Antarmuka *Foundry* Panel

Untuk masuk ke dalam *foundry* panel, pemain harus menekan tombol *foundry* pada tombol utama. Dalam *foundry* panel, pemain dapat meningkatkan senjata dan armor serta membeli *item*. Pemain bisa menggulir ke atas untuk melihat list barang yang dapat ditingkatkan atau dibeli. Terdapat 6 panel pada *foundry*. Pada sisi kiri setiap panel menampilkan ikon barang. Tepat disebelah gambar ikon terdapat nama barang dan keterangannya. Lalu pada sisi kanan terdapat nilai status yang bertambah atau dipulihkan serta tombol dengan harga didalamnya. Tombol tersebut dapat dipilih jika uang atau *gold* yang dimiliki pemain cukup untuk membeli atau meningkatkan barang. Antarmuka *foundry* panel dapat dilihat pada Gambar 4.9.

Gambar 4.9 Antarmuka *Foundry* Panel

Khusus untuk membeli *item*, akan muncul *pop up item quantity* untuk menentukan jumlah *item* yang akan dibeli. Pada panel *pop up* pemain dapat menambah jumlah *item* yang dibeli dengan menekan tombol panah ke kanan, dan untuk mengurangi dengan menekan tombol panah ke kiri. Pada sisi bawah panel terdapat tombol untuk membeli *item* tersebut. Di dalam tombol

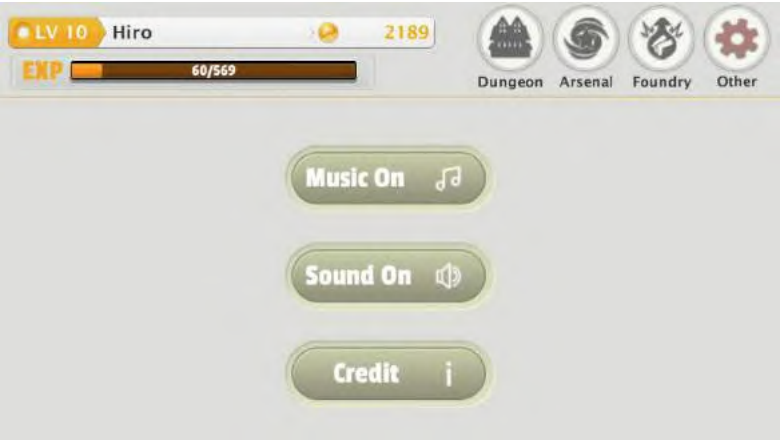
tersebut terdapat jumlah harga yang dibayar setelah menentukan jumlah *item* yang akan dibeli. Tombol lainnya adalah tombol *close* yang berada pada sisi kanan atas panel. Antarmuka *pop up item quantity* dapat dilihat pada Gambar 4.10.



Gambar 4.10 Antarmuka *Pop Up Item Quantity*

4.2.2.4 Implementasi Antarmuka *Other* Panel

Tombol terakhir pada tombol utama adalah *other*. Terdapat tiga tombol pada panel ini. Tombol pertama untuk mengaktifkan atau menonaktifkan musik, tombol kedua untuk mengaktifkan atau menonaktifkana suara, sedangkan untuk tombol terakhir untuk menampilkan pesan kredit dalam pembuatan permainan ini. Ketika pemain menekan tombol *credit*, akan muncul *pop up credit*. Seperti yang dijelaskan sebelumnya, *pop up credit* berisi kredit dalam pembuatan permainan ini. Pada *pop up credit* hanya ada satu tombol, yakni tombol close. Selama *pop credit* tampak di layar, pemain tidak dapat mengakses tombol yang lain. Antarmuka *other* panel dapat dilihat pada Gambar 4.11, sedangkan *pop up credit* ditampilkan pada Gambar 4.12.



Gambar 4.11 Antarmuka *Other* Panel

Gambar 4.12 Antarmuka *Pop Up Credit*

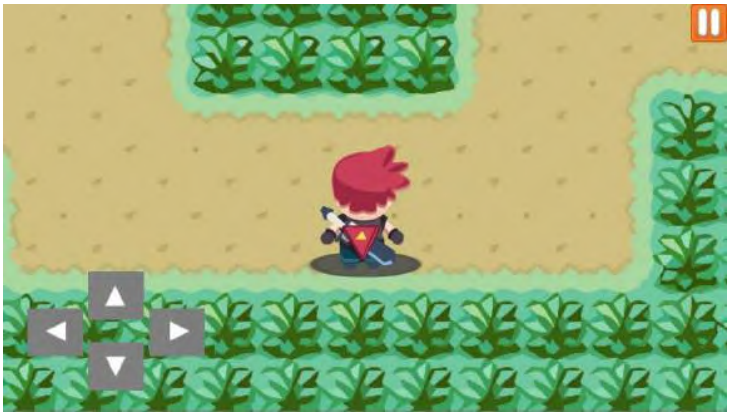
4.2.3 Implementasi Antarmuka *Dungeon*

Setelah mimilih *stage* yang diinginkan, pemain akan memasuki *dungeon*. Ketika pemain telah berada di *dungeon*, pemain akan melihat *pop up* pesan berupa *task list* yang harus dilakukan. *Pop up* tersebut tidak memiliki tombol, namun *pop up*

tersebut akan hilang dengan otomatis setelah 2 detik. Antarmuka *pop up* pesan dapat dilihat pada Gambar 4.13.

Gambar 4.13 Antarmuka *Pop Up* Pesan

Setelah *pop up* menghilang, pemain dapat mengakses tombol *on-screen*. Tombol *on-screen* pada *dungeon* berupa tombol 4 arah yang digunakan untuk menggerakkan karakter pemain. Tombol *on-screen* lainnya adalah tombol *pause* untuk menjeda permainan. Antarmuka *dungeon* dapat dilihat pada Gambar 4.14



Gambar 4.14 Antarmuka *Dungeon*

Ketika pemain menekan tombol *pause*, maka permainan akan dijeda dan muncul *pop up pause*. Pada *pop up* ini, terdapaat 3 tombol yang dapat digunakan. Tombol *resume* untuk memulai kembali permainan, tombol *exit dungeon* untuk keluar dari *dungeon*, dan tombol *task* untuk memunculkan kembali *pop up task list*. Antarmuka *pop up pause* dapat dilihat pada Gambar 4.15.

Gambar 4.15 Antarmuka *Pop Up Pause*

4.2.4 Implementasi Antarmuka *Battle*

Ketika pemain bertemu musuh di *dungeon*, pemain akan masuk ke dalam *battle mode*. Tampilan awal dari *battle mode* menampilkan 2 karakter beserta *status bar*-nya serta tiga tombol dibagaian bawah. Karakter pemain berada pada sisi kiri, sedangkan karakter musuh berada pada sisi kanan. *Status bar* berada di atas setiap karakter. Untuk karakter pemain, pada *status bar*-nya terdapat indikator *HP* pada bagian atas dan *MP* pada bagian bawah serta ikon karakter pada sisi sebelah kiri. Sedangkan untuk karakter musuh hanya menampilkan indikator *HP* dan ikon karakter. Untuk tombol pada bagian bawah, tombol paling kiri untuk menyerang musuh dengan serangan pedang, tombol di tengah untuk membuka panel *skill*, dan tombol bagian kanan untuk membuka panel *item*. Terdapat tombol jeda pada *battle* mode. Tombol tersebeut berada

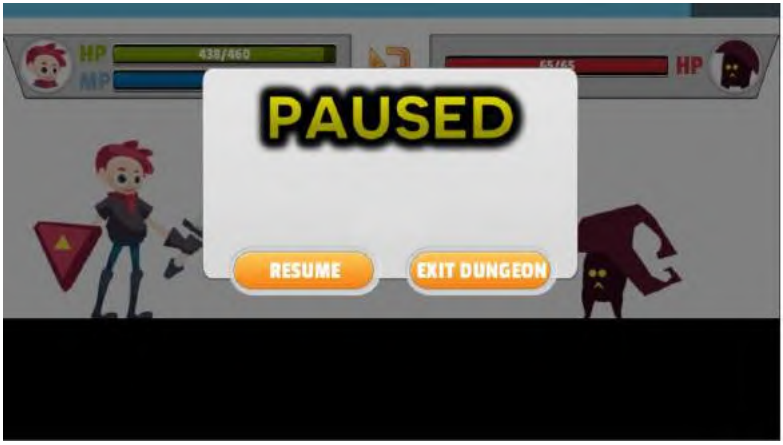
pada latar belakang permainan, sehingga pemain cukup menyentuh pada latar belakang saja. Antarmuka awal dari *battle mode* dapat dilihat pada Gambar 4.16.

Gambar 4.16 Antarmuka *Battle Mode*

Masuk ke dalam *attack* panel, terdapat beberap perubahan pada antarmuka dibanding antarmuka sebelumnya. Pertama pada bagian atas terdapat indikator waktu yang menandakan waktu yang tersisa untuk membentuk kata. Kemudian terdapat tombol diantara *status bar* masing-masing karakter, yakni tombol *refresh* yang digunakan untuk mengulang pemilihan huruf dari awal. Pada bagian bawah terdapat 8 tombol. 7 tombol pertama adalah tombol untuk memilih huruf. Tombol yang mewakili huruf dapat digunakan, sementara tombol yang tidak mewakili huruf tidak dapat digunakan. Tombol kedelepan digunakan untuk *submit* kata yang telah disusun. Ketika pemain menekan tombol *pause*, maka akan muncul *pop up pause*. Saat *pop up pause* aktif, pada bagian bawah permainan terdapat panel hitam yang berguna untuk mentupi huruf agar pemain tidak dapat melihat huruf saat permainan dijeda. Pada panel *pause* yang terletak di tengah terdapat 2 tombol, yakni tombol *resume* untuk melanjutkan permainan dan tombol *exit dungeon* untuk keluar ke *main menu.*

Antarmuka *attack mode* dapat dilihat pada Gambar 4.17, sedangkan antarmuka *pop up pause attack* dapat dilihat pada Gambar 4.18

Gambar 4.17 Antarmuka *Attack Mode*



Gambar 4.18 Antarmuka *Pop Up Pause Attack*

Untuk *skill* panel, antarmuka yang dimiliki hampir sama dengan antarmuka pada awal *battle mode*. Hanya saja, pada bagian

bawah terdapat 4 tombol. Tiga tombol awal adalah tombol untuk menggunakan *skill*, dimana tombol pertama untuk menggunakan *skill basic*, tombol kedua untuk menggunakan *skill buff*, dan tombol ketiga untuk menggunakan *skill ultimate*. Ikon settiap *skill* dapat berbeda, mengikuti elemen yang dipilih oleh pemain pada awal permainan. Sedangkan tombol terakhir atau tombol keempat adalah untuk kembali ke panel awal *battle mode*. Antarmuka *skill mode* dapat dilihat pada Gambar 4.19.

Gambar 4.19 Antarmuka *Skill Mode*

Untuk *item* panel, antarmuka yang dimiliki hampir sama dengan antarmuka pada awal *battle mode*. Hanya saja, pada bagian bawah terdapat 5 tombol. 4 tombol awal adalah tombol untuk menggunakan *item*, dimana tombol pertama untuk menggunakan *potion*, tombol kedua untuk menggunakan *super potion*, tombol ketiga untuk menggunakan *magic potion*, dan tombol keempat untuk menggunakan *super magic potion*. Pada bagian bawah ikon *item* terdapat angka yang menandakan jumlah *item* yang dimiliki. Sedangkan tombol terakhir atau tombol kelima adalah untuk kembali ke panel awal *battle mode*. Antarmuka *item mode* dapat dilihat pada Gambar 4.20.

Gambar 4.20 Antarmuka *Item Mode*

4.3 Implementasi Algoritma Permainan

Dalam menerapkan algoritma kedalam permainan, diperlukan beberapa tahap antara lain proses *filtering*, proses pemilihan kata, proses pengumpulan kata, serta proses pengecekan kata.

4.3.1 Implementasi Proses *Filtering*

Pada tahap ini, kata yang berasal dari kamus referensi akan dipisah menjadi 26 data, dimana setiap data akan mewakili huruf depan. Sebelum memisah menjadi 26 bagian, kata yang akan dimasukkan hanyalah kata dengan panjang tiga sampai 7 huruf saja. Kemudian, setiap huruf akan diubah menjadi huruf kapital agar proses pengecekan lebih mudah.Tujuan dari memisahkan kata berdasar huruf depan adalah agar mempercepat dan mempermudah proses pencarian. Proses ini satu-satunya yang dilakukan dengan menggunakan bahasa C++ dengan IDE CodeBlocks. Proses *filtering* dapat dilihat pada Kode Sumber 4.1.

```
1.  int main () {
2.    int wordLength;
3.    string line;
```

```
4.    ifstream myfile ("words2s.txt");
5.    ofstream myfilter ("filter_a.txt");
6.
7.    if (myfile.is_open())
8.    {
9.      while ( getline (myfile,line) )
10.     {
11.         transform(line.begin(), line.end(),
    line.begin(), ptr_fun<int, int>(toupper));
12.         wordLength = line.length();
13.         if (wordLength >= 3 && wordLength <=7){
14.             if (line[0] == 'A' || line[0] == 'a' ){
15.                 if (myfilter.is_open()){
16.                     myfilter << line << "\n";
17.                 }
18.             }
19.         //filter sampai huruf Z
20.         }
21.     }
22.     myfilter.close();
23.     myfile.close();
24.   }
25.   else cout << "Unable to open file";
26.   return 0;
27. }
```

Kode Sumber 4.1 Proses *Filtering*

4.3.2 Implementasi Proses Pemilihan Kata

Setelah membagi kamus referensi menjadi 26 bagian, tahap selanjutnya adalah memilih kata secara acak. Dengan dipilihnya kata acak dari kamus referensi, maka setiap putaran pada permainan nanti akan memiliki minimal satu solusi jawaban. Pemilihan kata ini ditentukan oleh pemilihan huruf awal secara acak. Setiap huruf awal memiliki bobot yang telah dijabarkan pada Tabel 3.12 pada Bab 3. Setelah mendapat huruf awal, sistem akan mengambil jumlah kata pada data tersebut dan mulai memilih kata secara acak. Proses ini dapat dilihat pada Kode Sumber 4.2.

```
1.  public string RandomRange(int random){
```

```
2.      int rangeA = 6;
3.          //set range huruf dari A sampai Z
4.      int rangeZ = 100;
5.
6.      int index;
7.      string lineWord = ".";
8.      int randomIndex;
9.
10.     if (random <= rangeA){
11.             getWords = kamusA.ToString ();
12.             kamusWord = getWords.Split (new
    char[]{'\n'});
13.             index = kamusWord.Length;
14.             randomIndex = Random.Range(1,index);
15.             lineWord = kamusWord[randomIndex];
16.     }
17.     return lineWord;
18. }
```

Kode Sumber 4.2 Proses Pemilihan Kata Acak

4.3.3 Implementasi Pengumpulan Solusi Kata

Pada tahap ini sistem akan menemukan kata yang dapat dibentuk dari kata acak sebelumnya menggunakan Algoritma *Backtracking*. Tahap pertama yang dilakukan adalah membuat akar dari setiap huruf yang ada pada kata acak. Setiap akar dari huruf mewakili data dari 26 bagian kamus referensi. Kemudian dibuat sebuah array penanda untuk mengetahui apakah sistem telah melewati cabang atau simpul dari akar yang dibentuk. Proses ini ditampilkan pada Kode Sumber 4.3.

```
1.  public void startBacktrack(string getLine){
2.      int n = getLine.Length;
3.      bool[] lineValue;
4.      lineValue = new bool[7];
5.      string newLine;
6.
7.      Array.Clear(listCorrectWord, 0,
    listCorrectWord.Length);
8.      indexList = 0;
```

```
9.
10.     for (int i=0;i<n;i++)
11.             lineValue[i] = true;
12.
13.     for (int i=0;i<n;i++) {
14.             tmpIndex = 0;
15.             newLine = String.Empty;
16.             newLine += getLine[i];
17.             lineValue[i] = false;
18.
19.             if(getLine[i] == 'A'){
20.                     getWords = kamusA.ToString();
21.                     kamusWord = getWords.Split (new
    char[]{'\n'});
22.             }
23.             //cek sampai huruf Z
24.             bck(0, n, getLine, lineValue, newLine,
    kamusWord);
25.             for (int j=0; j<n; j++)
26.                     lineValue[j] = true;
27.
28.             getWords = String.Empty;
29.             Array.Clear(kamusWord, 0,
    kamusWord.Length);
30.     }
31. }
```

Kode Sumber 4.3 Pembuatan Akar Setiap Huruf

Tahap selanjutnya adalah membentuk cabang atau simpul. Sistem akan megecek apakah simpul sudah terlewati atau belum. Jika belum, maka sistem akan melewati simpul tersebut. Ketika melewati simpul, sistem akan memasukkan huruf yang ada pada simpul tersebut dan mengeceknya, sebagai contoh huruf pada akar adalah ‘A’ dan huruf pada simpul adalah ‘L’. Kemudian sistem akan mengecek apakah ada kata dengan awal ‘AL’. Jika ada, maka simpul ‘L’ akan diteruskan hingga menemukan kata yang benar, dan jika tidak ada, maka simpul ‘L’ akan diputus dan dilanjutkan dengan simpul selanjutnya jika ada. Ketika mengecek, jika kata yang diperoleh pada simpul lebih dari tiga dan terdapat kata yang

sama pada kamus referensi dan panjang katanya sama, maka kata yang diperoleh dari simpul dimasukkan kedalam suatu *array of string* sebagai kumpulan solusi yang telah ditemukan. Dalam pengecekan akan dilihat pula jika terdapat kata yang sama pada simpul berbeda, sebagai contoh pada simpul pertama membentuk kata 'DOO' dan simpul ketiga membentuk kata 'DOO' juga, maka simpul ketiga akan diputus dan tidak diteruskan lagi karena pencarian selanjutnya akan sama dengan simpul pertama. Pencarian dinyatakan telah berakhir jika seluruh akar telah diiterasi. Dalam pembentukan simpul atau cabang dijabarkan pada Kode Sumber 4.4, sedangkan pengecakan kata dijabarkan pada Kode Sumber 4.5.

```
1.  void bck(int y, int n, string line, bool[] lineValue,
    string text, string[] myreferensi){
2.      int cek;
3.      for(int j=0;j<n;j++){
4.              if(lineValue[j]!=false){
5.                      lineValue[j]=false;
6.                      text += line[j];
7.                      y++;
8.                      cek =
    Checking(text,j,myreferensi);
9.                      if (cek == 0)
10.
        bck(y,n,line,lineValue,text,myreferensi);
11.                     lineValue[j]=true;
12.                     text = text.Remove (y);
13.                     y--;
14.             }
15.     }
16. }
```

Kode Sumber 4.4 Pembuatan Simpul

```
1.  private int Checking(string line, int i, string[]
    listWord){
2.      int n = line.Length;
3.      string tmpListWord;
```

```
4.      if(n>0){          //pencegahan kata yg sama
5.              if(tmpCek[n-2] != line)
6.                      tmpCek[n-2] = line;
7.              else if(tmpCek[n-2] == line)
8.                      return -1;
9.      }
10.     int x = listWord.Length;
11.
12.     for (int j = tmpIndex; j<x; j++){
13.             tmpListWord = String.Empty;
14.             tmpListWord = listWord[j];
15.             tmpListWord =
    tmpListWord.Remove(tmpListWord.Length-1);
16.             if ( n > tmpListWord.Length){
17.                     for ( int y = tmpListWord.Length;
    y<n; y++)
18.                             tmpListWord += ".";
19.             }
20.
21.             if(line[n-1] == tmpListWord[n-1] &&
    line[n-2] == tmpListWord[n-2]){
22.                     if(n>2){
23.                     if (line == tmpListWord){
24.                             listCorrectWord[indexList]
    = tmpListWord;
25.                                     indexList++;
26.                             }
27.                     }
28.                     tmpIndex = j;
29.                     return 0;
30.             }
31.             else if (tmpListWord[n-1] > line[n-1] ||
    tmpListWord[n-2] > line[n-2]){
32.                     tmpIndex = j;
33.                     return -1;
34.             }
35.             else if(n >=4 && tmpListWord.Length >=4){
36.                     if(tmpListWord[n-3] > line[n-3] ||
    tmpListWord[n-4] > line[n-4]){
37.                             tmpIndex = j;
38.                             return -1;
39.                     }
```

```
40.             }
41.     }
42.     return 0;
43. }
```

Kode Sumber 4.5 Pengecekan dan Pembuatan Solusi

4.3.4 Implementasi Pengecekan Kata pada Permainan

Proses pengecekan kata yang dilakukan ketika pemain memberikan jawaban dari huruf yang diacak menggunakan Algoritma *Brute Force*. Sistem akan mengecek satu persatu kata yang diberikan oleh pemain hingga sebanyak kata yang terdapat pada solusi. Jika terdapat kata yang sesuai dengan solusi, maka pemain dapat menyerang, jika tidak maka pemain gagal dalam menyerang. Proses ini dijabarkan pada Kode Sumber 4.6.

```
1.  private void PlayerAttack(){
2.      int foundWord = 0;
3.      string[] getListWord;
4.      getListWord = backtrackingController.SetListWord
    ();
5.      int n = getListWord.Length;
6.      for (int i=0; i<n; i++) {
7.              if (collectWord == getListWord[i]){
8.                      foundWord = 1;
9.                      break;
10.             }
11.     }
12. }
```

Kode Sumber 4.6 Pengecekan Kata

BAB V
PENGUJIAN DAN EVALUASI

Pada bab ini akan dilakukaan pengujian dan evalusai terhadap implementasi yang telah dilakukan.

5.1 Lingkungan Pengujian

Lingkungan pelaksanaan uji coba meliputi perangkat keras dan perangkat lunak yang akan digunakan pada sistem ini. Lingkungan uji coba yang digunakan adalah sebuah *smartphone* dengan spesifikasi ditunjukkan pada Tabel 5.1.

Tabel 5.1 Lingkungan Uji Coba

Perangkat Keras	Tipe: Prosesor: Memori: GPU:	Lenovo Vibe X2 MediaTek MT6595M 2048MB RAM PowerVR G600
Perangkat Lunak	Sistem Operasi:	Android Lollipop 5.0

5.2 Skenario Pengujian

Pada skenario pengujian, proses pengujian dilakukan menggunakan metode *black-box* berdasarkan skenario yang telah ditentukan terhadap fungsionalitas permainan dan pengujian dilakukan dengan survei langsung kepada pengguna.

5.2.1 Pengujian Fungsionalitas

Pengujian fungsionalitas dilakukan untuk menguji apakah fungsionalitas yang diidentifikasi pada tahap kebutuhan benar-benar diimplementasikan dan bekerja semestinya. Pengujian ini juga berdasarkan dari rumusan masalah yang telah dijabarkan pada Subbab 1.1. Selain itu akan disiapkan beberapa skenario pengujian sebagai tolak ukur keberhasilan pengujian. Pengujian fungsionalitas dilakukan dengan menggunakan metode *black-box.*

5.2.1.1 Pengujian Mencari Musuh

Pengujian mencari musuh merupakan pengujian terhadap permainan untuk menemukan musuh yang berada di *dungeon* dan ketika bertemu dengan musuh akan menampilkan *battle mode*. Skenario pengujian ini dapat dilihat pada Tabel 5.2.

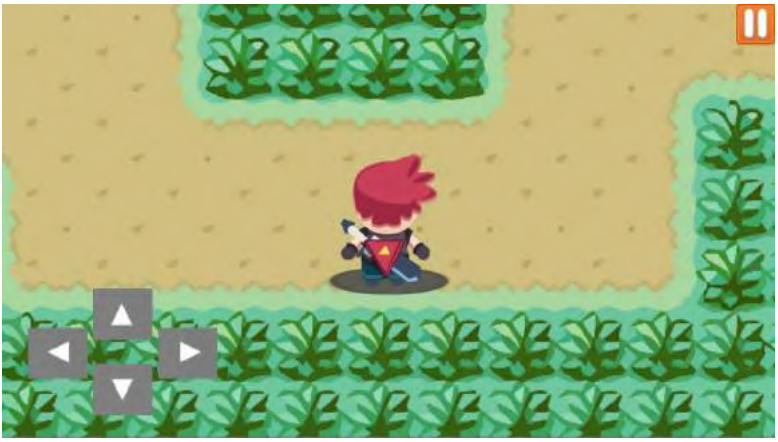
Tabel 5.2 Pengujian Mencari Musuh

ID	UJ-P-01
Nama	Mencari Musuh
Tujuan	Pengguna dapat menemukan musuh yang tersebar di *dungeon*
Kondisi awal	Pengguna sudah berada pada *main menu* untuk memulai permainan
Skenario	Pengguna memilih *dungeon stage* pada *dungeon map*, kemudian pengguna berkeliling mengelilingi *dungeon* untuk mencari musuh
Keluaran yang diharapkan	Pengguna menemukan musuh yang tersembunyi di dalam *dungeon*
Hasil uji coba	Berhasil
Kondisi akhir	Pengguna berhasil menemukan musuh dan masuk ke *battle mode*

Gambar 5.1 Kondisi Awal pada *Main Menu*

Kondisi awal pengguna sudah berada pada *main menu*, dimana *default* panelnya adalah *dungeon* panel yang ditunjukkan pada Gambar 5.1. Untuk skenario memilih *dungeon stage* dan berkeliling *dungeon* ditampilkan pada Gambar 5.2 dan Gambar 5.3. Sementara kondisi akhir ketika bertemu musuh ditampilkan pada Gambar 5.4.

Gambar 5.2 Memilih *Dungeon Stage*



Gambar 5.3 *Explore Dungeon*

Gambar 5.4 Masuk *Battle Mode*

5.2.1.2 Pengujian Pengacakan dan Pengecekan Kata

Pengujian pengecekan kata merupakan pengujian terhadap sistem permainan dalam membentuk solusi kata dan mengecek input dari penggunan. Skenario pengujian ini dapat dilihat pada Tabel 5.3.

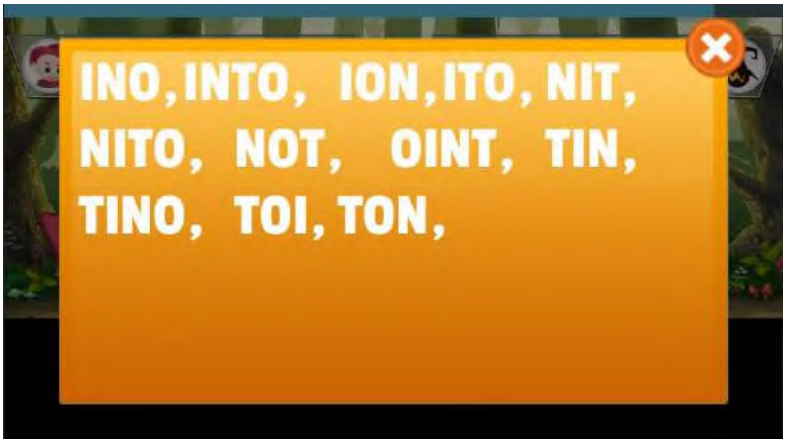
Tabel 5.3 Pengujian Pengacakan dan Pengecekan Kata

ID	UJ-P-02
Nama	Pengacakan dan Pengecekan Kata
Tujuan	Sistem mampu mengacak kata baru setiap putaran dan dapat membuat solusi kata berdasarkan kamus referensi dan melakukan pengecekan kata yang telah diinputkan oleh pengguna
Kondisi awal	Pengguna berada pada *attack panel* dan bersiap menyusun kata
Skenario 1	Pengguna menyusun kata dengan benar sebelum batas waktu berakhir
Skenarion 2	Pengguna menyusun kata yang tidak tepat sebelum batas waktu berakhir

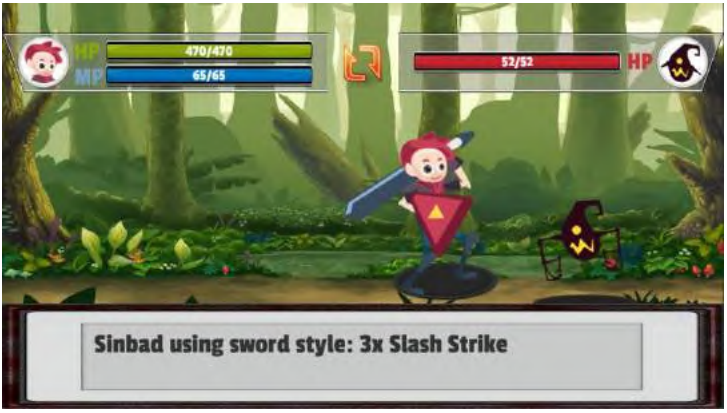
Keluaran yang diharapkan	Akan terbentuk kumpulan huruf dari kata yang terpilih secara acak dan terdapat solusi dari setiap kata acak serta sistem mampu mengenali input kata dari pengguna
Hasil uji coba	Berhasil
Kondisi akhir	Setiap putaran menghasilkan kata acak baru dan terbentuk solusi dari setiap kata acak serta sistem dapat menentukan benar salah dari input kata pengguna

Kondisi awal pengguna sudah berada pada *attack panel*, dimana telah muncul kumpulan huruf dari kata yang terpilih secara acak yang ditunjukkan pada Gambar 5.5. Dari kata acak yang terpilih, sistem membentuk kumpulan solusi kata yang dapat dibentuk. Solusi kata tersebut ditampilkan pada Gambar 5.6. Sesuai dengan skenario pertama, pengguna membentuk salah satu kata yang benar, yakni "ION" dan karakter pemain berhasil menyerang musuh. Bukti karakter pemain dapat menyerang musuh ditampilkan pada Gambar 5.7.

Gambar 5.5 Kondisi Awal *Attack Panel*



Gambar 5.6 Kumpulan Solusi yang Dibentuk



Gambar 5.7 Pemain Berhasil Menyerang

Untuk skenario kedua, sistem akan membentuk solusi kata dari kata acak baru pada putaran selanjutnya yang ditampilkan pada Gambar 5.8. Kemudian pengguna membentuk kata yang salah, yakni "HRS" dan pemain gagal menyerang seperti yang ditampilkan pada Gambar 5.9. Selama pengujian ini berlangsung, didapati solusi kata berupa singkatan seperti "USH". Hal ini

dikarenakan kata yang berasal dari *Infochimps* terdapat singkatan-singkatan yang umum digunakan oleh orang asing.

Gambar 5.8 Kumpulan Solusi Untuk Skenario 2

Gambar 5.9 Pemain Gagal Menyerang

5.2.1.3 Pengujian Strategi Melawan Musuh

Pengujian strategi melawan musuh merupakan pengujian dalam melawan musuh dengan beberapa cara yang telah disediakan

oleh system permainan. Skenario pengujian ini dapat dilihat pada Tabel 5.4.

Tabel 5.4 Pengujian Strategi Melawan Musuh

ID	UJ-P-03
Nama	Strategi Melawan Musuh
Tujuan	Pengguna dapat menentukan pilihan dalam menyusun strategi
Kondisi awal	Pengguna berada dalam *battle mode*
Skenario 1	Pengguna menyerang dengan menyusun kata
Skenarion 2	Pengguna menyerang musuh dengan menggunakan *skill* 1 dan 3
Skenario 3	Pengguna meningkatkan kekuatan dengan menggunakan *skill* 2
Skenario 4	Pengguna memulihkan kondisi *HP* dan *MP* menggunakan *item potion*
Keluaran yang diharapkan	Pengguna mampu mengatur strategi dalam mengalahkan musuh
Hasil uji coba	Berhasil
Kondisi akhir	Pengguna mampu mengalahkan musuh dengan strategi yang telah diatur

Pada skenario satu, pengguna dapat menyerang musuh dengan menyusun kata pada huruf acak yang muncul pada layar. Pengujian menyerang dengan menyusun kata telah dijabarkan pada Subbab 5.2.1.2. Pengguna dapat memilih *skill* yang dimiliki untuk menyerang musuh atau memperkuat karakter pemain. Pemilihan *skill* ditunjukkan pada Gambar 5.10. Untuk skenario 2, pengguna dapat menggunakan *skill* 1 atau *skill* 3 untuk menyerang musuh. *Skill* 1 merupakan *skill basic* dengan penggunaan *MP* yang tidak besar, sedangkan *skill 3* merupakan *skill* tingkat akhir yang mengonsumsi cukup banyak *MP*. Penggunaan *skill* 1 dapat dilihat pada Gambar 5.11, sementara penggunaan *skill* 3 dapat dilihat pada Gambar 5.12.

Gambar 5.10 Pemilihan *Skill*

Gambar 5.11 Penggunaan *Skill* 1

Penggunaan *skill* 1 maupun *skill* 3 dapat memberikan efek kepada musuh. Musuh yang terkene efek dari dua *skill* tersebut menyebabkan musuh tidak dapat bergerak. Jika musuh terkena efek *stun*, ditampilkan pada Gambar 5.13.

Gambar 5.12 Penggunaan *Skill* 3

Gambar 5.13 Musuh Terkena Efek *Stun*

Untuk skenario 3, pemain dapat menggunakan *skill* 2 untuk menaikkan status dasar karakter pemain. Penggunaan *skill* 2 dapat dilihat pada Gambar 5.14.

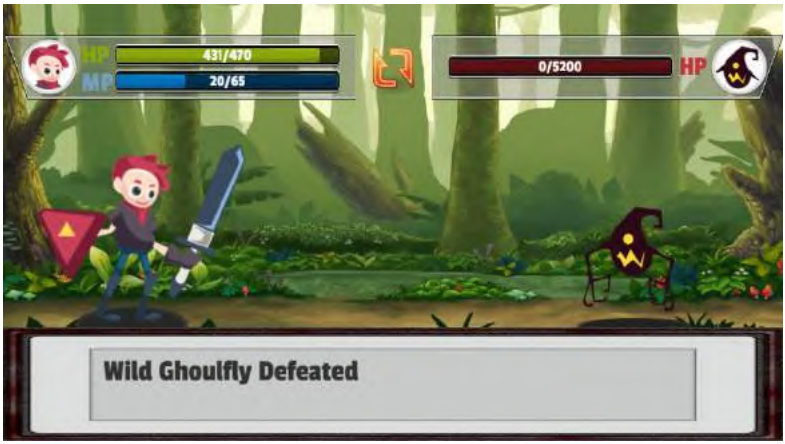
Gambar 5.14 Penggunaan *Skill* 2

Untuk skenario 4, pemain dapat memulihkan kondisi *HP* maupun *MP* dengan menggunakan *item potion*. Penggunaan *Healing Potion* ditunjukkan pada Gambar 5.15, sedangkan penggunaan *Magic Potion* ditunjukkan pada Gambar 5.16.

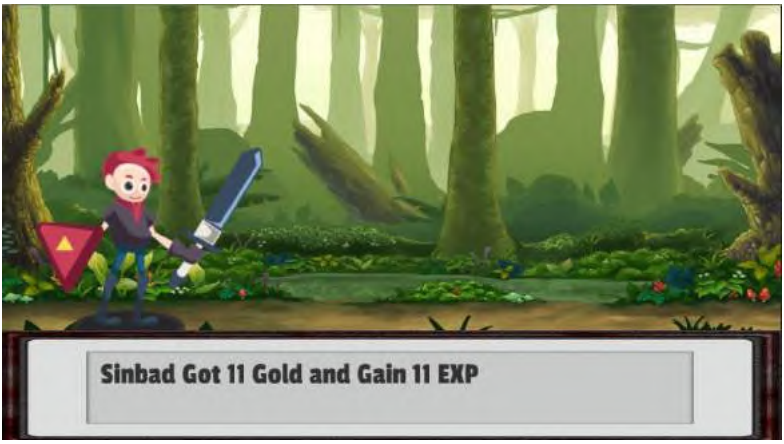
Gambar 5.15 Penggunaan *Healing Potion*

Gambar 5.16 Penggunaan *Magic Potion*

Dengan strategi yang dijabarkan diatas, pengguna dapat mengalahkan musuh. Musuh kalah jika *HP* musuh kurang dari 0. Musuh yang kalah ditampilkan pada Gambar 5.17. Setiap kali pengguna mengalahkan musuh, maka pengguna akan mendapat *EXP* dan *gold*. *EXP* dan *gold* yang diperoleh pengguna dapat dilihat pada Gambar 5.18



Gambar 5.17 Musuh Kalah



Gambar 5.18 Pengguna Mendapat *Reward*

5.2.2 Pengujian Terhadap Pengguna

Permainan ini perlu diuji oleh pengguna secara langsung. Hal ini bertujuan untuk mengetahui tingkat kenyamanan dan kemudahan, serta menyenangkan dalam memainkannya. Tujuan lainnya adalah untuk menilai daya tarik pengguna awam terhadap permainan dengan *genre RPG*. Pengujian ini mengacu pada metode *blackbox* yang telah dijabarkan dalam menguji fungsionalitas aplikasi.

Pengujian dilakukan oleh beberapa orang yang gemar bermain *game RPG* dan beberapa orang yang tidak gemar bermain *game RPG*. Selain itu kriteria peserta pengujian juga berdasarkan kemampuan bahasa Inggris pengguna yang mengetahui kosakata bahasa Inggris dengan jumlah banyak maupun dengan jumlah terbatas. Pengguna akan diminta untuk mencoba menjalankan permainan terlebih dahulu dan kemudian pengguna diminta untuk mengisi kuesioner yang terlah disediakan. Kuesioner dapat dilihat pada Tabel 5.5. Setelah mengisi kuesioner, pengguna diminta untuk memberikan kritik dan saran terhadap permainan ini. Bobot penilaian pada kuesioner memiliki ketentuan sebagai berikut:

- Sangat Setuju (SS) = 5
- Setuju (S) = 4
- Cukup (C) = 3
- Tidak Setuju (TS) = 2
- Sangat Tidak Setuju (STS) = 1

Tabel 5.5 Kuesioner Pengguna

No.	Parameter	Penilaian				
		STS	TS	C	S	SS
	Antarmuka					
1	Aplikasi memiliki tampilan dan desain yang menarik					
2	Aplikasi memiliki tampilan yang sesuai dengan tema permainan *RPG*					
3	Aplikasi memiliki menu yang mudah digunakan					
4	Aplikasi memiliki tata letak tombol yang sesuai					
	Performa dan Kenyamanan					
5	Aplikasi nyaman untuk dimainkan					
6	Aplikasi menampilkan animasi dengan lancar					
7	Aplikasi menampilkan perpindahan menu dengan lancar					
8	Aplikasi mampu menampilkan dan mengecek kata dengan cepat					

	Materi Permainan					
9	Saya merasa mudah dalam membentuk kata					
10	Kata Bahasa Inggris yang dapat dibentuk adalah valid					
11	Saya tertarik untuk memainkan permainan ini					
12	Saya merasa tidak bosan dengan *gameplay* permainan ini					

5.3 Evaluasi Pengujian

Tahap evaluasi dibagi menjadi dua bagian, yaitu evaluasi pengujian fungsionalitas dan evaluasi pengujian aplikasi terhadap pengguna.

5.3.1 Evaluasi Pengujian Fungsionalitas

Rekap mengenai hasil pengujian fungsionalitas dapat dilihat pada Tabel 5.6. Berdasarkan data pada tabel tersebut, dapat disimpulkan bahwa semua skenario pengujian berhasil dijalankan. Sehingga dapat disimpulkan bahwa fungsionalitas dari aplikasi telah bekerja sesuai dengan yang diharapkan.

Tabel 5.6 Hasil Pengujian Fungsionalitas

No.	ID	Nama	Hasil
1	UJ-P-01	Mencari Musuh	Berhasil
2	UJ-P-02	Pengacakan dan Pengecekan Kata	Berhasil
3	UJ-P-03	Strategi Melawan Musuh	Berhasil

5.3.2 Evaluasi Pengujian Terhadap Pengguna

Pengisian kuesioner dilakukan oleh 10 orang Mahasiswa Teknik Informatika ITS. Rentang umur pengguna berkisar antara 20 sampai 23 tahun. Pengguna terdiri dari 1 wanita dan 9 pria. Hasil kuesioner dari setiap kriteria penilaian akan diambil modusnya. Jawaban kuesioner dapat dilihat pada lembar Lampiran. Sementara hasil kuesioner dapat dilihat pada Tabel 5.7.

Tabel 5.7 Hasil Kuesioner

No.	Parameter	Penilaian				
		STS	TS	C	S	SS
	Antarmuka					
1	Aplikasi memiliki tampilan dan desain yang menarik	-	-	1	**6**	3
2	Aplikasi memiliki tampilan yang sesuai dengan tema permainan *RPG*	-	-		**5**	**5**
3	Aplikasi memiliki menu yang mudah digunakan	-	1	3	**5**	1
4	Aplikasi memiliki tata letak tombol yang sesuai	-	-	2	**6**	2
	Performa dan Kenyamanan					
5	Aplikasi nyaman untuk dimainkan	-	-	1	**9**	-
6	Aplikasi menampilkan animasi dengan lancar	-	1	1	**4**	**4**
7	Aplikasi menampilkan perpindahan menu dengan lancar	-	**4**	1	3	2
8	Aplikasi mampu menampilkan dan mengecek kata dengan cepat	-	-	1	4	**5**
	Materi Permainan					
9	Saya merasa mudah dalam membentuk kata	1	1	2	**5**	1
10	Kata Bahasa Inggris yang dapat dibentuk adalah valid	-	-	2	**5**	3

11	Saya tertarik untuk memainkan permainan ini	-	-	1	**5**	4
12	Saya merasa tidak bosan dengan *gameplay* permainan ini	-	-	**4**	3	3

5.3.2.1 Evaluasi Antarmuka

Untuk parameter antarmuka, terdapat 4 aspek penilaian. Aspek pertama adalah tampilan dan desain. Dari aspek ini, mayoritas pengguna setuju bahwa permainan ini memiliki tampilan dan desain yang menarik dengan nilai 6 dari 10 pengguna yang memilihnya. Aspek berikutnya adalah kesesuaian tampilan dengan tema *game RPG*. Nilai dari aspek ini seimbang, dimana 5 dari 10 pengguna setuju bahwa tampilan sudah sesuai dan 5 pengguna lainya sangat setuju dengan kesesuaian tampilan dengan tema permainan, sehingga dapat disimpulkan bahwa permainan ini memiliki tampilan yang sesuai dengan tema permainan *RPG*. Aspek selanjutnya adalah kemudahan menu, dimana 5 dari 10 pengguna setuju bahwa permainan memiliki menu yang mudah digunakan. Namun, 1 orang pengguna tidak setuju menu mudah digunakan karena tidak terbiasa dengan penamaan menunya. Untuk aspek terakhir yakni tata letak tombol. Mayoritas pengguna setuju bahwa permainan memiliki tata letak tombol yang sesuai. Rincian hasil kuesioner bisa dilihat pada Tabel 5.7.

5.3.2.2 Evaluasi Performa dan Kenyamanan

Untuk parameter performa dan kenyamanan, akan dilihat bagaimana performa serta kelancaran dari permainan selama pengujian. Aspek pertama adalah kenyamanan. Dari aspek ini, mayoritas pengguna setuju bahwa permainan ini nyaman dimainkan dengan nilai 9 dari 10 pengguna yang memilihnya. Aspek berikutnya adalah kelancaran animasi. Nilai dari aspek ini seimbang, dimana 4 dari 10 pengguna setuju dan 4 pengguna lainya sangat setuju bahwa animasi yang ditampilkan lancar, sehingga dapat disimpulkan bahwa permainan ini menampilkan animasi dengan lancar. Hanya saja, 1 pengguna tidak setuju bahwa aplikasi

berjalan lancar, karena menganggap animasi saat berjalan masih patah-patah. Aspek selanjutnya adalah perpindahan menu, dimana 4 dari 10 pengguna tidak setuju bahwa permainan menampilkan perpindahan menu dengan lancar. Pengguna tersebut merasa terganggu dengan waktu yang cukup lama untuk berpindah dari *dungeon scene* ke *battle scene*. Namun, 3 pengguna setuju serta 2 pengguna lainnya juga sangat setuju bahwa perpindahan menu sudah lancar. Untuk aspek terakhir yakni kecepatan menampilkan dan pengecekan kata. Mayoritas pengguna sangat setuju bahwa permainan mampu menampilkan dan mengecek kata dengan cepat dengan nilai 5 dari 10 pengguna memilihnya. Rincian hasil kuesioner bisa dilihat pada Tabel 5.7.

5.3.2.3 Evaluasi Materi Permainan

Untuk parameter materi permainan, terdapat dua aspek yang dipengaruhi oleh kemampuan bahasa Inggris pengguna. Untuk mengetahui kemampuan bahasa Inggris seorang pengguna, parametenya adalah kemampuan pengguna untuk mengetahui kosakata bahasa Inggris sebanyak 1000 kata. Selama pengujian, terdapat 5 orang yang mengetahui lebih dari 1000 kosakata dan 5 orang yang mengetahui kurang dari 1000 kosakata. Hasil kuesioner untuk aspek kemudahan membentuk kata dan validasi kata bahasa Inggris bagi pengguna yang mengetahui lebih dari 1000 kosakata dapat dilihat pada Tabel 5.8, sedangkan untuk pengguna yang mengetahui kurang dari 1000 kosakata dapat dilihat pada Tabel 5.9.

Tabel 5.8 Hasil Kuesioner untuk Pengguna dengan Kemampuan Mengetahui Lebih Dari 1000 Kosakata

No.	Parameter	Penilaian				
		STS	TS	C	S	SS
1	Saya merasa mudah dalam membentuk kata	1	-	-	**3**	1
2	Kata Bahasa Inggris yang dapat dibentuk adalah valid	-	-	-	**3**	2

Tabel 5.9 Hasil Kuesioner untuk Pengguna dengan Kemampuan Mengetahui Kurang Dari 1000 Kosakata

No.	Parameter	Penilaian				
		STS	TS	C	S	SS
1	Saya merasa mudah dalam membentuk kata	-	1	**2**	**2**	-
2	Kata Bahasa Inggris yang dapat dibentuk adalah valid	-	-	**2**	**2**	1

Dari hasil yang ditunjukkan pada Tabel 5.8 dan Tabel 5.9, terdapat beberapa perbedaan. Untuk aspek kemudahan dalam membentuk kata, pengguna yang mengetahui lebih dari 1000 kosakata mayoritas setuju bahwa dirinya mudah dalam membentuk kata, dibandingkan dengan pengguna yang hanya mengetauhi kurang dari 1000 kosakata, dimana mayoritas pengguna tersebut merasa cukup mudah dalam menyusun kata. Perlu diperhatikan terdapat satu pengguna dengan kemampun mengenali lebih dari 1000 kosakata yang tidak setuju dengan kemudahan dalam menyusun kata, dikarenakan waktu yang terdapat saat bermain membuat pengguna tersebut panik sehingga susah untuk menyusun kata. Untuk aspek berikutnya, yakni validasi kata bahasa Inggris. Mayoritas pengguna baik dengan kemampuan bahasa Inggris yang baik maupun yang kurang, menyatakan setuju bahwa kata bahasa Inggris yang dapat dibentuk adalah valid. Namun, dari keseluruhan pengguna, hanya 3 orang yang sangat setuju bahwa bahasa Inggris yang dapat dibentuk adalah valid. Hal ini dikarenakan ketika pengguna menguji permainan ini, ditemukan nama seperti "KYLE" dan singkatan seperti "ABA (*Applied Behavior Analysis*)" dan menganggap kata tersebut bukanlah kata bahasa Inggris yang valid. Hanya saja untuk pengguna dengan kemampuan bahasa Inggris lebih menoleransi kata tersebut dengan menilai lebih baik dibanding pengguna dengan kemampuan bahasa Inggris yang kurang.

Untuk aspek selanjutnya, yakni ketertarikan bermain, mayoritas pemain setuju untuk memainkan kembali permainan ini

dengan nilai 5 dari 10 pengguna yang memilihnya dan 4 dari 10 pengguna cenderung sangat setuju untuk memainkan permainan ini kembali. Sedangkan untuk aspek *fun*, mayoritas pengguna menganggap cukup tidak bosan dengan *gameplay* pada permainan ini dimana 4 dari 10 pengguna yang menyatakan hal tersebut. Namun 6 pengguna lainnya setuju bahwa *gameplay* permainan ini tidak membosankan dan bahkan 3 di antaranya sangat setuju dengan *gameplay* seperti ini. Rincian hasil kuesioner bisa dilihat pada Tabel 5.7.

LAMPIRAN

KUESIONER TUGAS AKHIR – 5112100010 FAJAR SETIAWAN

RANCANG BANGUN RANDOM WORD GENERATOR DENGAN ALGORITMA BACKTRACKING PADA GAMEPLAY UNTUK GAME AMBROSIA

Identitas Responden

Nama Lengkap : Tri Sutrisno Nusantara Usia : 21 Tahun
Pekerjaan : Mahasiswa Jenis Kelamin : (L) P

A. KARAKTERISTIK RESPONDEN

1. Apakah Anda gemar bermain *game* dengan genre *RPG*?
   a. Ya (b.) Tidak
2. Seberapa sering Anda memainkan *game RPG* dalam 1 minggu?
   a. Tidak Pernah (c.) 2 kali e. 4 kali
   b. 1 kali d. 3 kali f. >= 5 kali
3. Apakah Anda mengetahui 1000 kosakata dalam bahasa Inggris?
   (a.) Ya b. Tidak

B. PENILAIAN TERHADAP APLIKASI

Isilah tabel dibawah ini dengan menggunakan tanda (√)

SS = Sangat Setuju *S = Setuju* *C = Cukup*
TS = Tidak Setuju *STS = Sangat Tidak Setuju*

No	Parameter Antarmuka	STS	TS	C	S	SS
1	Aplikasi memiliki tampilan dan desain yang menarik				√	
2	Aplikasi memiliki tampilan yang sesuai dengan tema permainan *RPG*				√	
3	Aplikasi memiliki menu yang mudah digunakan			√		
4	Aplikasi memiliki tata letak tombol yang sesuai				√	
	Parameter Performa dan Kenyamanan					
5	Aplikasi nyaman untuk dimainkan			√		
6	Aplikasi menampilkan animasi dengan lancar				√	
7	Aplikasi menampilkan perpindahan menu dengan lancar		√			
8	Aplikasi mampu menampilkan dan mengecek kata dengan cepat				√	
	Parameter Materi Permainan					
9	Saya merasa mudah dalam membentuk kata	√				
10	Kata Bahasa Inggris yang dapat dibentuk adalah valid				√	
11	Saya tertarik untuk memainkan permainan ini				√	
12	Saya merasa tidak bosan dengan *gameplay* permainan ini			√		

C. KRITIK DAN SARAN

Transisi ketika battle ditambahkan supaya bisa menunggu, game balancing diperbaiki soalnya dari awal sudah susah

Surabaya, 8 juni 2016

Tri Sutrisno Nusantara

Gambar A.1 Kuesioner Responden Pertama

KUESIONER TUGAS AKHIR – 5112100010 FAJAR SETIAWAN

RANCANG BANGUN RANDOM WORD GENERATOR DENGAN ALGORITMA BACKTRACKING PADA GAMEPLAY UNTUK GAME AMBROSIA

Identitas Responden

Nama Lengkap : Ikram Aulia Fahdi Usia : 22 Tahun
Pekerjaan : Pelajar Siswa Jenis Kelamin : (L) / P

A. KARAKTERISTIK RESPONDEN

1. Apakah Anda gemar bermain *game* dengan genre *RPG*?
   (a) Ya b. Tidak
2. Seberapa sering Anda memainkan *game RPG* dalam 1 minggu?
   a. Tidak Pernah c. 2 kali e. 4 kali
   (b) 1 kali d. 3 kali f. >= 5 kali
3. Apakah Anda mengetahui 1000 kosakata dalam bahasa Inggris?
   (a) Ya b. Tidak

B. PENILAIAN TERHADAP APLIKASI

Isilah tabel dibawah ini dengan menggunakan tanda (√)

SS = Sangat Setuju *S = Setuju* *C = Cukup*
TS = Tidak Setuju *STS = Sangat Tidak Setuju*

No	Parameter Antarmuka	STS	TS	C	S	SS
1	Aplikasi memiliki tampilan dan desain yang menarik				√	
2	Aplikasi memiliki tampilan yang sesuai dengan tema permainan *RPG*				√	
3	Aplikasi memiliki menu yang mudah digunakan				√	
4	Aplikasi memiliki tata letak tombol yang sesuai			√		
	Parameter Performa dan Kenyamanan					
5	Aplikasi nyaman untuk dimainkan				√	
6	Aplikasi menampilkan animasi dengan lancar				√	
7	Aplikasi menampilkan perpindahan menu dengan lancar			√		
8	Aplikasi mampu menampilkan dan mengecek kata dengan cepat					√
	Parameter Materi Permainan					
9	Saya merasa mudah dalam membentuk kata				√	
10	Kata Bahasa Inggris yang dapat dibentuk adalah valid					√
11	Saya tertarik untuk memainkan permainan ini					√
12	Saya merasa tidak bosan dengan *gameplay* permainan ini				√	

C. KRITIK DAN SARAN

Kelemahannya ditemukan yang, kemunculan monster, item dan map
Kalau bisa tombol untuk menggerakkan monster diganti tombol
bundar agar mempermudah pemain dalam menggerakkan
pemain

Surabaya, 08 – 06 – 2016

Ikram AF

Gambar A.2 Kuesioner Responden Kedua

KUESIONER TUGAS AKHIR – 5112100010 FAJAR SETIAWAN

RANCANG BANGUN RANDOM WORD GENERATOR DENGAN ALGORITMA BACKTRACKING PADA GAMEPLAY UNTUK GAME AMBROSIA

Identitas Responden

Nama Lengkap : Rahma Fida Fadhilah Usia : 21 Tahun
Pekerjaan : Mahasiswa Jenis Kelamin : L (P)

A. KARAKTERISTIK RESPONDEN

1. Apakah Anda gemar bermain *game* dengan genre *RPG*?
   (a) Ya b. Tidak
2. Seberapa sering Anda memainkan *game RPG* dalam 1 minggu?
   a. Tidak Pernah c. 2 kali e. 4 kali
   b. 1 kali d. 3 kali (f.) >= 5 kali
3. Apakah Anda mengetahui 1000 kosakata dalam bahasa Inggris?
   a. Ya (b.) Tidak

B. PENILAIAN TERHADAP APLIKASI

Isilah tabel dibawah ini dengan menggunakan tanda (√)

SS = Sangat Setuju *S = Setuju* *C = Cukup*
TS = Tidak Setuju *STS = Sangat Tidak Setuju*

No	Parameter Antarmuka	STS	TS	C	S	SS
1	Aplikasi memiliki tampilan dan desain yang menarik				√	
2	Aplikasi memiliki tampilan yang sesuai dengan tema permainan *RPG*					√
3	Aplikasi memiliki menu yang mudah digunakan			√		
4	Aplikasi memiliki tata letak tombol yang sesuai				√	
	Parameter Performa dan Kenyamanan					
5	Aplikasi nyaman untuk dimainkan				√	
6	Aplikasi menampilkan animasi dengan lancar					√
7	Aplikasi menampilkan perpindahan menu dengan lancar		√			
8	Aplikasi mampu menampilkan dan mengecek kata dengan cepat					√
	Parameter Materi Permainan					
9	Saya merasa mudah dalam membentuk kata				√	
10	Kata Bahasa Inggris yang dapat dibentuk adalah valid					√
11	Saya tertarik untuk memainkan permainan ini					√
12	Saya merasa tidak bosan dengan *gameplay* permainan ini					√

C. KRITIK DAN SARAN

Terdapat bug saat lambang victory muncul, kemudian tidak sengaja layar tertekan, muncul menu kemudian aplikasi stop atau tidak berjalan. Tambahkan minimap agar pemain tahu dimana ia berada sekarang

Surabaya, 8 Juni 2016

RAHMA FIDA FADHILAH

Gambar A.3 Kuesioner Responden Ketiga

KUESIONER TUGAS AKHIR – 5112100010 FAJAR SETIAWAN

RANCANG BANGUN RANDOM WORD GENERATOR DENGAN ALGORITMA BACKTRACKING PADA GAMEPLAY UNTUK GAME AMBROSIA

Identitas Responden

Nama Lengkap : Dimas Widdy — Usia : 22 Tahun
Pekerjaan : Mahasiswa — Jenis Kelamin : (L) / P

A. KARAKTERISTIK RESPONDEN

1. Apakah Anda gemar bermain *game* dengan genre *RPG*?
   (a) Ya b. Tidak
2. Seberapa sering Anda memainkan *game RPG* dalam 1 minggu?
   a. Tidak Pernah (c) 2 kali e. 4 kali
   b. 1 kali d. 3 kali f. >= 5 kali
3. Apakah Anda mengetahui 1000 kosakata dalam bahasa Inggris?
   a. Ya (b) Tidak

B. PENILAIAN TERHADAP APLIKASI

Isilah tabel dibawah ini dengan menggunakan tanda (√)

SS = Sangat Setuju S = Setuju C = Cukup
TS = Tidak Setuju STS = Sangat Tidak Setuju

No	Parameter Antarmuka	STS	TS	C	S	SS
1	Aplikasi memiliki tampilan dan desain yang menarik					√
2	Aplikasi memiliki tampilan yang sesuai dengan tema permainan *RPG*				√	
3	Aplikasi memiliki menu yang mudah digunakan				√	
4	Aplikasi memiliki tata letak tombol yang sesuai				√	
	Parameter Performa dan Kenyamanan					
5	Aplikasi nyaman untuk dimainkan				√	
6	Aplikasi menampilkan animasi dengan lancar					√
7	Aplikasi menampilkan perpindahan menu dengan lancar				√	
8	Aplikasi mampu menampilkan dan mengecek kata dengan cepat				√	
	Parameter Materi Permainan					
9	Saya merasa mudah dalam membentuk kata			√		
10	Kata Bahasa Inggris yang dapat dibentuk adalah valid			√		
11	Saya tertarik untuk memainkan permainan ini				√	
12	Saya merasa tidak bosan dengan *gameplay* permainan ini			√		

C. KRITIK DAN SARAN

Tampilan fase waktunya tolak dekat kemenangan. kadang susah membentuk kata.

Surabaya, 8 Juni 2016

Dimas Widdy

Gambar A.4 Kuesioner Responden Keempat

KUESIONER TUGAS AKHIR – 5112100010 FAJAR SETIAWAN

RANCANG BANGUN RANDOM WORD GENERATOR DENGAN ALGORITMA BACKTRACKING PADA GAMEPLAY UNTUK GAME AMBROSIA

Identitas Responden

Nama Lengkap : Angga Saputra Usia : 22 Tahun
Pekerjaan : Mahasiswa Jenis Kelamin : (L)/P

A. KARAKTERISTIK RESPONDEN

1. Apakah Anda gemar bermain *game* dengan genre *RPG*?
   (a.) Ya b. Tidak
2. Seberapa sering Anda memainkan *game RPG* dalam 1 minggu?
   a. Tidak Pernah c. 2 kali e. 4 kali
   (b.) 1 kali d. 3 kali f. >= 5 kali
3. Apakah Anda mengetahui 1000 kosakata dalam bahasa Inggris?
   (a.) Ya b. Tidak

B. PENILAIAN TERHADAP APLIKASI

Isilah tabel dibawah ini dengan menggunakan tanda (√)

SS = Sangat Setuju *S = Setuju* *C = Cukup*
TS = Tidak Setuju *STS = Sangat Tidak Setuju*

No	Parameter Antarmuka	STS	TS	C	S	SS
1	Aplikasi memiliki tampilan dan desain yang menarik					√
2	Aplikasi memiliki tampilan yang sesuai dengan tema permainan *RPG*					√
3	Aplikasi memiliki menu yang mudah digunakan		√			
4	Aplikasi memiliki tata letak tombol yang sesuai					√
	Parameter Performa dan Kenyamanan					
5	Aplikasi nyaman untuk dimainkan				√	
6	Aplikasi menampilkan animasi dengan lancar					√
7	Aplikasi menampilkan perpindahan menu dengan lancar		√			
8	Aplikasi mampu menampilkan dan mengecek kata dengan cepat				√	
	Parameter Materi Permainan					
9	Saya merasa mudah dalam membentuk kata					√
10	Kata Bahasa Inggris yang dapat dibentuk adalah valid					√
11	Saya tertarik untuk memainkan permainan ini					√
12	Saya merasa tidak bosan dengan *gameplay* permainan ini					√

C. KRITIK DAN SARAN

- Seharusnya ada loading screen. sehingga user mengetahui apakah proses sedang berjalan atau crash. (jangan black screen)
- [redacted] pilih bahasa terlebih dahulu untuk menampilkan tutorial.

Surabaya, 8 Juni 2016

ANGGA

Gambar A.5 Kuesioner Responden Kelima

KUESIONER TUGAS AKHIR – 5112100010 FAJAR SETIAWAN

RANCANG BANGUN RANDOM WORD GENERATOR DENGAN ALGORITMA BACKTRACKING PADA GAMEPLAY UNTUK GAME AMBROSIA

Identitas Responden

Nama Lengkap : Dimas R.H Usia : 23 Tahun
Pekerjaan : Mahasiswa Jenis Kelamin (L)/P

A. KARAKTERISTIK RESPONDEN

1. Apakah Anda gemar bermain *game* dengan genre *RPG*?
   (a) Ya b. Tidak
2. Seberapa sering Anda memainkan *game RPG* dalam 1 minggu?
   (a) Tidak Pernah c. 2 kali e. 4 kali
   b. 1 kali d. 3 kali f. >= 5 kali
3. Apakah Anda mengetahui 1000 kosakata dalam bahasa Inggris?
   a. Ya (b) Tidak

B. PENILAIAN TERHADAP APLIKASI

Isilah tabel dibawah ini dengan menggunakan tanda (√)
SS = Sangat Setuju S = Setuju C = Cukup
TS = Tidak Setuju STS = Sangat Tidak Setuju

No	Parameter Antarmuka	STS	TS	C	S	SS
1	Aplikasi memiliki tampilan dan desain yang menarik					√
2	Aplikasi memiliki tampilan yang sesuai dengan tema permainan *RPG*					√
3	Aplikasi memiliki menu yang mudah digunakan				√	
4	Aplikasi memiliki tata letak tombol yang sesuai				√	
	Parameter Performa dan Kenyamanan					
5	Aplikasi nyaman untuk dimainkan				√	
6	Aplikasi menampilkan animasi dengan lancar				√	
7	Aplikasi menampilkan perpindahan menu dengan lancar		√			
8	Aplikasi mampu menampilkan dan mengecek kata dengan cepat			√		
	Parameter Materi Permainan					
9	Saya merasa mudah dalam membentuk kata		√			
10	Kata Bahasa Inggris yang dapat dibentuk adalah valid			√		
11	Saya tertarik untuk memainkan permainan ini			√		
12	Saya merasa tidak bosan dengan *gameplay* permainan ini			√		

C. KRITIK DAN SARAN

Lebih di pertimbangkan leveling pada pemilihan kata.

Surabaya, 8 Juni 2016

Dimas Rizkahadi

Gambar A.6 Kuesioner Responden Keenam

KUESIONER TUGAS AKHIR – 5112100010 FAJAR SETIAWAN

RANCANG BANGUN RANDOM WORD GENERATOR DENGAN ALGORITMA BACKTRACKING PADA GAMEPLAY UNTUK GAME AMBROSIA

identitas Responden

Nama Lengkap : Ignatius Abraham S. Usia : 22 Tahun
Pekerjaan : Mahasiswa Jenis Kelamin : (L)/P

A. KARAKTERISTIK RESPONDEN

1. Apakah Anda gemar bermain *game* dengan genre *RPG*?
   a. Ya b. Tidak
2. Seberapa sering Anda memainkan *game RPG* dalam 1 minggu?
   a. Tidak Pernah c. 2 kali e. 4 kali
   b. 1 kali d. 3 kali >= 5 kali
3. Apakah Anda mengetahui 1000 kosakata dalam bahasa Inggris?
   a. Ya b. Tidak

B. PENILAIAN TERHADAP APLIKASI

Isilah tabel dibawah ini dengan menggunakan tanda (v)

SS = Sangat Setuju S = Setuju C = Cukup
TS = Tidak Setuju STS = Sangat Tidak Setuju

No	Parameter Antarmuka	STS	TS	C	S	SS
1	Aplikasi memiliki tampilan dan desain yang menarik			✓		
2	Aplikasi memiliki tampilan yang sesuai dengan tema permainan *RPG*				✓	
3	Aplikasi memiliki menu yang mudah digunakan			✓		
4	Aplikasi memiliki tata letak tombol yang sesuai			✓		
	Parameter Performa dan Kenyamanan					
5	Aplikasi nyaman untuk dimainkan				✓	
6	Aplikasi menampilkan animasi dengan lancar		✓			
7	Aplikasi menampilkan perpindahan menu dengan lancar				✓	
8	Aplikasi mampu menampilkan dan mengecek kata dengan cepat					✓
	Parameter Materi Permainan					
9	Saya merasa mudah dalam membentuk kata				✓	
10	Kata Bahasa Inggris yang dapat dibentuk adalah valid				✓	
11	Saya tertarik untuk memainkan permainan ini				✓	
12	Saya merasa tidak bosan dengan *gameplay* permainan ini			✓		

C. KRITIK DAN SARAN

Untuk lebih menarik, menambahkan animasi skill, perpindahan scene dungeon dengan battle, memperingan game.

Surabaya, 8 Juni 2016

Egnatius Abraham Susanto

Gambar A.7 Kuesioner Responden Ketujuh

KUESIONER TUGAS AKHIR – 5112100010 FAJAR SETIAWAN

RANCANG BANGUN RANDOM WORD GENERATOR DENGAN ALGORITMA BACKTRACKING PADA GAMEPLAY UNTUK GAME AMBROSIA

Identitas Responden

Nama Lengkap : Rahmat Irfan Usia : 20 Tahun
Pekerjaan : Mahasiswa Jenis Kelamin : (L) P

A. KARAKTERISTIK RESPONDEN

1. Apakah Anda gemar bermain *game* dengan genre *RPG*?
   (a.) Ya b. Tidak
2. Seberapa sering Anda memainkan *game RPG* dalam 1 minggu?
   a. Tidak Pernah c. 2 kali e. 4 kali
   b. 1 kali d. 3 kali (f.) >= 5 kali
3. Apakah Anda mengetahui 1000 kosakata dalam bahasa Inggris?
   a. Ya (b.) Tidak

B. PENILAIAN TERHADAP APLIKASI

Isilah tabel dibawah ini dengan menggunakan tanda (√)

SS = Sangat Setuju *S = Setuju* *C = Cukup*
TS = Tidak Setuju *STS = Sangat Tidak Setuju*

No	Parameter Antarmuka	STS	TS	C	S	SS
1	Aplikasi memiliki tampilan dan desain yang menarik				√	
2	Aplikasi memiliki tampilan yang sesuai dengan tema permainan *RPG*					√
3	Aplikasi memiliki menu yang mudah digunakan					√
4	Aplikasi memiliki tata letak tombol yang sesuai					√
	Parameter Performa dan Kenyamanan					
5	Aplikasi nyaman untuk dimainkan				√	
6	Aplikasi menampilkan animasi dengan lancar					√
7	Aplikasi menampilkan perpindahan menu dengan lancar					√
8	Aplikasi mampu menampilkan dan mengecek kata dengan cepat				√	
	Parameter Materi Permainan					
9	Saya merasa mudah dalam membentuk kata				√	
10	Kata Bahasa Inggris yang dapat dibentuk adalah valid				√	
11	Saya tertarik untuk memainkan permainan ini					√
12	Saya merasa tidak bosan dengan *gameplay* permainan ini					√

C. KRITIK DAN SARAN

Kualitas tombol pada menu ditambahkan

Surabaya, 16 Juni 2016

Rahmat Irfan

Gambar A.8 Kuesioner Responden Kedelapan

KUESIONER TUGAS AKHIR – 5112100010 FAJAR SETIAWAN

RANCANG BANGUN RANDOM WORD GENERATOR DENGAN ALGORITMA BACKTRACKING PADA GAMEPLAY UNTUK GAME AMBROSIA

Identitas Responden

Nama Lengkap : Luthfi F. Soehadak Usia : 22 Tahun
Pekerjaan : Mahasiswa Jenis Kelamin : (L) P

A. KARAKTERISTIK RESPONDEN

1. Apakah Anda gemar bermain *game* dengan genre *RPG*?
   (a.) Ya b. Tidak
2. Seberapa sering Anda memainkan *game RPG* dalam 1 minggu?
   a. Tidak Pernah c. 2 kali e. 4 kali
   b. 1 kali d. 3 kali (f.) >= 5 kali
3. Apakah Anda mengetahui 1000 kosakata dalam bahasa Inggris?
   (a.) Ya b. Tidak

B. PENILAIAN TERHADAP APLIKASI

Isilah tabel dibawah ini dengan menggunakan tanda (√)

SS = Sangat Setuju *S = Setuju* *C = Cukup*
TS = Tidak Setuju *STS = Sangat Tidak Setuju*

No	Parameter Antarmuka	STS	TS	C	S	SS
1	Aplikasi memiliki tampilan dan desain yang menarik				√	
2	Aplikasi memiliki tampilan yang sesuai dengan tema permainan *RPG*				√	
3	Aplikasi memiliki menu yang mudah digunakan				√	
4	Aplikasi memiliki tata letak tombol yang sesuai				√	
	Parameter Performa dan Kenyamanan					
5	Aplikasi nyaman untuk dimainkan				√	
6	Aplikasi menampilkan animasi dengan lancar				√	
7	Aplikasi menampilkan perpindahan menu dengan lancar				√	
8	Aplikasi mampu menampilkan dan mengecek kata dengan cepat					√
	Parameter Materi Permainan					
9	Saya merasa mudah dalam membentuk kata				√	
10	Kata Bahasa Inggris yang dapat dibentuk adalah valid				√	
11	Saya tertarik untuk memainkan permainan ini				√	
12	Saya merasa tidak bosan dengan *gameplay* permainan ini				√	

C. KRITIK DAN SARAN

- Map dungeon diperbanyak macamnya untuk menghindari kebosanan

Surabaya, 10 Juni 2016

Gambar A.9 Kuesioner Responden Kesembilan

KUESIONER TUGAS AKHIR – 5112100010 FAJAR SETIAWAN

RANCANG BANGUN RANDOM WORD GENERATOR DENGAN ALGORITMA BACKTRACKING PADA GAMEPLAY UNTUK GAME AMBROSIA

Identitas Responden

Nama Lengkap : Rizky DS Usia : 22 Tahun
Pekerjaan : Mahasiswa Jenis Kelamin (L)/P

A. KARAKTERISTIK RESPONDEN

1. Apakah Anda gemar bermain *game* dengan genre *RPG*?
   a. Ya (b.) Tidak
2. Seberapa sering Anda memainkan *game RPG* dalam 1 minggu?
   a. Tidak Pernah c. 2 kali e. 4 kali
   (b.) 1 kali d. 3 kali f. >= 5 kali
3. Apakah Anda mengetahui 1000 kosakata dalam bahasa Inggris?
   a. Ya (b.) Tidak

B. PENILAIAN TERHADAP APLIKASI

Isilah tabel dibawah ini dengan menggunakan tanda (√)

SS = Sangat Setuju *S = Setuju* *C = Cukup*
TS = Tidak Setuju *STS = Sangat Tidak Setuju*

No	Parameter Antarmuka	STS	TS	C	S	SS
1	Aplikasi memiliki tampilan dan desain yang menarik				√	
2	Aplikasi memiliki tampilan yang sesuai dengan tema permainan *RPG*					√
3	Aplikasi memiliki menu yang mudah digunakan				√	
4	Aplikasi memiliki tata letak tombol yang sesuai				√	
	Parameter Performa dan Kenyamanan					
5	Aplikasi nyaman untuk dimainkan				√	
6	Aplikasi menampilkan animasi dengan lancar			√		
7	Aplikasi menampilkan perpindahan menu dengan lancar					√
8	Aplikasi mampu menampilkan dan mengecek kata dengan cepat					√
	Parameter Materi Permainan					
9	Saya merasa mudah dalam membentuk kata			√		
10	Kata Bahasa Inggris yang dapat dibentuk adalah valid				√	
11	Saya tertarik untuk memainkan permainan ini				√	
12	Saya merasa tidak bosan dengan *gameplay* permainan ini				√	

C. KRITIK DAN SARAN

1. Masih ada bug di tutorial ketika tutorial belum selesai semua
2. Tantangan dalam dungeon bisa lebih banyak.

Surabaya, 10 Juni 2016

Rizky Dwi S.

Gambar A.10 Kuesioner Responden Kesepuluh

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini dijelaskan mengenai kesimpulan yang dapat diambil dari hasil pengujian yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga diberi saran untuk pengembangan aplikasi kedepannya.

6.1. Kesimpulan

Dalam proses pengerjaan Tugas Akhir ini, mulai dari tahap analisis, desain, implementasi, hingga pengujian didapatkan kesimpulan sebagai berikut:

1. Berdasarkan pengujian terhadap pengacakan kata, permainan ini berhasil memilih kata secara acak berdasarkan huruf awal yang dibangun menggunakan *Game Engine* Unity.
2. Dari pengujian yang dilakukan, penggunaan Algoritma *Backtracking* mampu mengekstrasi solusi kata berdasarkan data kamus yang ada.
3. Selama pengujian pada pengacakan dan pengecekan kata, Infochimps menyajikan kosakata yang cukup banyak meskipun tidak terdapat kata jamak. Hanya saja, pada kosakata tersebut melibatkan singkatan dan nama dalam bahasa Inggris yang tidak familiar oleh kebanyakan orang Indonesia.
4. Pada pengujian yang dilakukan, terdapat kendala kenyamanan saat perpindahan *scene dungeon* dengan *scene battle*, yakni butuh waktu yang cukup lama dalam *load scene battle*. Hal ini dikarenakan sistem memerlukan banyak *memory* untuk memroses *asset* yang terdiri atas 1 *spritesheet* untuk 1 gerakan.
5. Dari pengujian yang telah dilakukan kepada pengguna, permainan ini cukup menarik dan cukup layak untuk dimainkan. Namun, banyak pengguna yang merasa kesulitan dalam membentuk kata, terutama bagi pengguna dengan pengetahuan kosakata yang terbatas.

6.2. Saran

Adapun saran yang diberikan untuk mengembangkan aplikasi kedepannya antara lain :

1. Mempertimbangkan kembali pengaturan desain level untuk awal permainan, khususnya jumlah monster dan pemilihan kata acak dibatasi.
2. Menambahkan suatu kamus yang terdiri atas kata berdasarkan tingkat kesulitannya untuk keseimbangaan level permaianan.
3. Menambahkan *loading scene* atau transisi saat perpindahan *scene* agar tidak menimbulkan kesan *lag* atau *crash*.
4. Meringankan *asset* serta beberapa proses untuk meningkatkan performa permainan.

DAFTAR PUSTAKA

[1] "Top 100 RPGs of All Time," IGN, [Online]. Available: http://www.ign.com/top/rpgs. [Accessed 9 Desember 2015].

[2] "Usability First - Methods - Usability Testing," Usability First, [Online]. Available: http://www.usabilityfirst.com/usability-methods/usability-testing/. [Accessed 13 Desember 2015].

[3] "Unity," Unity, [Online]. Available: https://unity3d.com/unity. [Accessed 8 Desember 2015].

[4] I. Sommerville, Software Engineering 9th edition, Buston: Pearson Education, 2011.

[5] "Download Android Studio and SDK Tools | Android Developers," Google Android, [Online]. Available: http://developer.android.com/sdk/index.html. [Accessed 14 Desember 2015].

[6] A. Levitin, "Backtracking," in *Introduction to The Design and Analysis of Algorithms*, New Jersey, Pearson Education, Inc, 2012, pp. 424-430.

[7] A. Levitin, "Brute Force," in *Introduction to The Design and Analysis of Algorithms*, New Jersey, Pearson Education, Inc, 2012, pp. 97-106.

[8] D. L. Adam Lipowski, "Roulette-wheel selection via stochastic acceptance," *Physica A*, vol. 391, pp. 2193-2196, 2012.

[9] "Code:Blocks," The Code::Blocks team, [Online]. Available: http://www.codeblocks.org/home. [Accessed 8 Desember 2015].

[10] "Infochimps: Big Data - Cloud Services," Infochimps, Inc., 23 Februari 2012. [Online]. Available: http://www.infochimps.com/datasets/word-list-350000-

simple-english-words-excel-readable. [Accessed 29 Januari 2016].

[11] "Scrabble | Word Games | Board Games | Scrabble Online," Hasbro, [Online]. Available: http://scrabble.hasbro.com/en-us/faq. [Accessed 5 Februari 2016].

[12] M. I. Assaat, "Aplikasi Algoritma Backtracking dalam Permainan Anagram," *MAKALAH IF2251 STRATEGI ALGORITMIK*, 2007.

BIODATA

Penulis bernama lengkap Fajar Setiawan lahir di Surabaya pada tanggal 29 Juli 1994, merupakan anak ke-2 dari 2 bersaudara. Penulis telah menempuh pendidikan formal pada SDN Kompleks IKIP I Makassar (2000-2006), SMPN 6 Makassar (2006-2009), SMAN 5 Makassar (2009), SMAN 4 Surakarta (2010-2012), dan S1 di jurusan Teknik Informatika, Institut Teknologi Sepuluh Nopember Surabaya, dengan rumpun mata kuliah Interaksi, Grafika, dan Seni (IGS).

Pengalaman berorganisasi penulis dimulai sejak memasuki bangku SMP. Organisasi pertama penulis adalah organisasi LSBN (Lingkungan Sekolah Bebas Narkoba) dan saat itu mendapat amanah menjadi Ketua LSBN. Pada jenjang SMA, penulis mengikuti Rohis dan masuk dalam divisi logistik. Selama masa perkuliahan, penulis kembali mengikuti organisasi kerohanian, yakni organisasi Keluarga Muslim Informatika (KMI). Pada tahun ke-2, penulis menjabat sebagai staf departemen syiar. Pada tahun ke-3, penulis menjabat sebagai staf ahli departemen sahada. Kritik dan saran sangat diharapkan guna meningkatkan kualitas dan penulisan selanjutnya. Kritik dan saran bisa disampaikan melalui email sfajar1507@gmail.com